# Table of Contents

1 Introduction .................................................. 3
   Who should read this typesetting guide? ........ 3
   What this guide does *not* cover ................. 4
   How to proceed .......................................... 4

2 The Basics .................................................. 6
   An acknowledgement ..................................... 6
   What’s so good about \TeX{} ............................ 6
   How the program works ................................ 9
   Staying in control ..................................... 13

3 \TeX{}nical Issues ......................................... 17
   Spacing .................................................... 17
   Grouping & parameters ................................. 18
   Paragraphs ................................................ 21
   Text vs. math ............................................ 22
   Ties & comments ........................................ 24

4 Summary of Commands ..................................... 26
   Ordinary text ............................................ 27
   Common symbols ....................................... 39
   Math formulas .......................................... 46
   Special effects ........................................ 62
   Document layout ...................................... 76

5 Appendices
   A. Reference Tables ................................... 86
   B. Working with Graphics ............................ 90
   C. Changes from Plain \TeX{} ......................... 92
   D. Other Resources .................................... 94
   E. Index of Commands .................................. 96
   F. Index of Topics ..................................... 99
1 Introduction

This guide introduces the \TeX{} typesetting system. It is intended for people with little or no experience, but it also serves as a reference for more advanced users. The first few chapters provide an overview of \TeX, while the latter ones describe specific typesetting commands. There are plenty of examples to study, hints for problem-writing, and quick reference tables. In short, there is something for everyone.

If you are reading about \TeX{} for the first time, you may not know that it is a general-purpose typesetting system. It is widely used for books, journal articles, professional papers and technical manuscripts. Most people run \TeX{} as a standalone program, but it also works very well as a typesetting engine or front-end to a database (the next chapter explains why). For that reason, it has been incorporated into Acces, EducAide’s test-making software.

Who should read this typesetting guide?

If you want to write material inside of Acces, then you need to learn a little about \TeX{}. It really does not take much effort to produce great-looking results; the knowledge and skills will come quickly. But methods for typesetting are different from ordinary typing and word-processing, so you cannot proceed without at least a general understanding of \TeX{} and, in particular, EducAide’s implementation of it.

You should read this guide if you are
- an Acces user, and you want to write your own problems or directions.
- a curriculum developer, and you want to create a database of problems.
- an experienced \TeX{} user, and you want to know more about EducAide’s format file.

Important: If you merely want to pick problems inside of Acces, then you do not need to know anything about \TeX{} and you can skip this guide entirely. There is no information here which is necessary for running Acces, selecting problems, or printing a document. In fact, you can put down this guide now and still enjoy nearly all of Acces’ features.
What this guide does not cover

Although TeX is an impressive and powerful program, the topics covered here are quite simple. The reason we can ignore more complicated ones is that Acces handles just about everything related to page layout. If you wanted to design your own page styles, then you would have to deal with TeX as a programming language and learn about such things as ‘boxes’ and ‘glue’. Fortunately, problem-writing has more to do with data than design—content instead of form—so we can concentrate on a few basic ideas.

If you want to know more about TeX, many other resources are available. Not only is the program discussed in books and journals, but, for the truly dedicated, there are even TeX classes at major universities. Thus, we tend to avoid topics that are covered elsewhere, and focus on what is unique to Acces. Perhaps you will become interested enough that you will want to learn more; in that case, you are encouraged to make use of other, more exhaustive guides. We hope to provide enough information to get you started and to make you comfortable with TeX, but not so much that it becomes overwhelming.

How to proceed

Before you go any further, we suggest you familiarize yourself with Acces. This means running the programs a few times, putting together documents, and experimenting with the Commands column on Acces’ main screen. Also, if you have not already done so, you should the chapter in the Reference Manual called Writing your own problems. It is not important that you know everything about the programs, but that you understand the concepts behind them, especially the idea of data being stored in a very general way.

- If you are new to TeX . . .
  We strongly recommend that you read the next part of this guide, called The basics. Then you can skim through the reference sections and look up particular commands when you want to learn more. Please note that all commands are indexed at the back of this guide, and math symbols are summarized in Appendix A.

- If you have some experience . . .
  You may want to re-familiarize yourself with the basics, and then concentrate on commands that are unique to Acces. Also, if you have used an older version of the program, you are encouraged to look at the section called document layout, which describes many new commands.
If you have a lot of experience...

You may want to know how EducAide’s format file differs from others, like Plain \TeX and \LaTeX. You will find this information in Appendix C. Keep in mind, also, that EducAide licenses an advanced version of the typesetter, called VTEX. Your Reference Manual explains how to run VTEX, so you can format, preview and print your own files.

Finally, no matter what your level of experience, we direct your attention to Appendix D, called Other Resources. There you will discover a wide range of support options, so you can get the most out of the typesetting system. EducAide’s technical support number is also listed, should any of your questions go unanswered.
The Basics

An acknowledgement

The \TeX\ typesetting system was developed by Professor Donald Knuth at Stanford University, with support from the National Science Foundation, American Mathematical Society, and other organizations. Knuth, a highly respected computer scientist and mathematician, devoted more than ten years to the project. When \TeX\ was completed in 1982, he graciously made the source code available to the public. Until 1990, when he announced his retirement from the project, Knuth added features, fixed bugs, and offered guidance to the growing community of \TeX\ users. Anyone who finds value in the program, and especially those who have make use of the source code, owe a great debt to its author. EducAide is no exception; our appreciation of Knuth’s work is acknowledged here.

What’s so good about \TeX?\n
\TeX\ has been adopted by universities, corporations, and individuals around the world for the production of books, articles and other kinds of printed material. Many people argue it is the best typesetting system around, the standard against which others should be judged. In addition to its precise handling of text, including foreign languages, the program is perfectly suited for mathematics and other technical subjects. For this reason, \TeX\ has been called “the de facto standard for scholarly papers in math and physics.” [Seybold Report on Desktop Publishing, September, 1992].

Nowadays, various word processors, desktop publishers, and equation editors accomplish some of the same things as \TeX. But the latter still has advantages which set it apart. Here are some of them:

- **Real typesetting**
  
  Anyone familiar with \TeX\ will tell you that its output is superior to commercial or off-the-shelf products. Its quality is evidenced by the fact that many large publishers have moved away from their proprietary typesetting systems and adopted \TeX. If you teach mathematics, science or engineering, it is very likely that the program was used in the production of your books and journals—if not by a publisher, then perhaps by an author.
Intelligent formatting

\( \text{\LaTeX} \) reads ahead and makes decisions based on what it encounters. Among other things, it can process conditional text; that is, it can change the wording (or layout) of an item in certain situations. As an example, consider several test items that share a picture. \( \text{\LaTeX} \) is smart enough to print the picture only once and put in references to it later on. This is what makes \( \text{\LaTeX} \) an ideal front-end to a database, because one does not know in advance which items will be selected, or how they will be arranged on a page.

Better line- and page-breaking

\( \text{\LaTeX} \) employs sophisticated algorithms for breaking lines and pages. (This is also true for mathematical formulas.) Consider what a word processor does: it breaks lines as soon as they get filled up—i.e., when it is convenient to do so. \( \text{\LaTeX} \) waits until the end of a paragraph and chooses optimal breakpoints for all of the preceding lines. The result is evenly spaced words and balanced paragraphs, regardless of line length, font size, etc.

Complete programmability

You will never have to do any programming, but it’s nice to know that \( \text{\LaTeX} \) can be customized for different applications. Acces, a test-making tool, is a good example of this. Right out of the box, \( \text{\LaTeX} \) knows nothing about how lessons and exams should be typeset. So EducAide has programmed \( \text{\LaTeX} \) to handle Acces’ documents. The wonderful part of all this is that you could re-program it to produce an entirely different style of document, if you desired. The methods for doing so are not discussed here, but Appendix D lists other sources of information.

Open-endedness

This idea follows from \( \text{\LaTeX} \)’s programmability. While it is true that the typesetter possesses a rich set of features, you can go beyond them and add your own. For instance, the Acces package includes commands for scrambling multiple choice answers. This is not a native feature of \( \text{\LaTeX} \), but sort of an add-on. Since \( \text{\LaTeX} \) can run other programs when it is typesetting a document, one can imagine linking it to graphics utilities, random problem generators, symbolic manipulators, even optical mark readers. EducAide is working on some of these things now.
A world-wide following

\TeX\ has been implemented on virtually every type of computer, from PCs to mainframes, and translated to dozens of languages. There are users’ groups in North America, Europe and Asia. One reason it has such a loyal following is that users know they control the program’s operation and, in some sense, they own it (since the code is public). Sure, \TeX\ adherents include a good number of computer whizzes, but also lovers of fine typography, researchers in the humanities, linguists, and so on.

---

### Signs of quality

What distinguishes book printing (typesetting) from ordinary printing? Two things to look for are kerning and ligatures, both of which \TeX\ handles automatically.

- Kerning is the process of moving certain letters closer together, so as to improve the appearance of words. The difference, though subtle, is a mark of high-quality printing. You can probably spot the difference in the words WAVE and WAVE (kerning is turned off in the second case).

- Ligatures are combinations of letters that are treated as one unit. Some common ligatures are ‘ff’, ‘fi’, and ‘fl’, which replace the separate letters ff, fi and fl. Professional typesetters are trained to insert ligatures, because the separate letters clash in most styles of type. You can see this when you compare the words ‘find’ and ‘find’.

---

8 Typesetting Guide
How the program works

\TeX\ can be thought of best as a document processor. In contrast to a word processor, which receives all of its input from the keyboard, \TeX\ reads and formats an entire document, usually in the form of a file.* A typical file contains a mix of typesetting commands and data (the stuff to be typeset). Although files may be specially prepared for \TeX, the reverse is also true: \TeX\ may be programmed to handle data that is already structured a particular way.

As \TeX\ typesets or formats a document, it creates what is known as a DVI (device-independent) file. This, in turn, gets processed by some other program, usually a printer driver or screen previewer. In other words, a document goes through two stages before you see any results. The explanation for this is simple: new device drivers can be added as necessary. When \TeX\ is churning away, figuring out line breaks, scaling pictures, etc., it does not care whether the document will eventually be previewed, printed, faxed or turned into digitized speech; it typesets the document just the same.

The next two pages contain before and after pictures of an Acces document. The before picture is just the contents of a file that is created by Acces and read by \TeX. The after picture shows the typeset results, obtained from a laser printer. Keep in mind that the file Acces creates is not in some required form. The structure is simply a convenient way of describing a lesson or exam. Other \TeX\ files may appear very different.

In case you are interested, here is a little more information about Acces’ \TeX\ file. The first part may be thought of as a file header. It contains document-wide commands, or parameters. You may recognize some of them as options from the Acces program; the names generally correspond to fields in dialog boxes. After the parameters, you will see miscellaneous pieces of information, like directories (to help \TeX\ locate files), user-defined fields (such as class name), and global commands (if any).

The real action takes place after \begindocument. It is followed by a series of \begindirection...\enddirection and \beginproblem...\endproblem commands, which form boundaries around items in the document. You will also notice some \dirno and \workspace commands, which pass along information from Acces’ main screen. Then everything is concluded with \enddocument. By the way, the only significant difference between a test, worksheet, calendar, or other type of document is the very first command, appropriately named \documenttype.

*\TeX\ can also receive input directly from the keyboard, but this never occurs when running Acces.
\documentclass=2
\basefonttype=0
\basefontsize=11pt
\firstheadID=3
\probcolumns=2
\spacingtype=2
\partlabelID=3
\setclass Algebra 2A
\settitle Test on quadratic equations
\setteacher Ms. Openshaw
\setdate 6/21/99
\begindocument
\beginmemo
Show all work for credit. Good luck!
\endmemo
\dirno=15
\begindirections
Solve.
\enddirections
\workspace=.5in
\beginproblem
(1+c)(13+c)=0
\endproblem
\beginproblem
0=p^2-6p-55
\endproblem
\beginproblem
The difference of two numbers is 6. The sum of their squares is 116. Find the numbers.
\endproblem
\beginproblem
A piece of wood is 16 in wide and 30 in long. How wide a strip must be cut off one end and one side to make the area of the wood 275 sq in?
\endproblem
\beginproblem
A rectangular patio is surrounded on three sides by a fence (the remaining side is up against the house). If the area of the patio is 150 ft squared, and the total length of fence is 35 ft, what is the length and width of the patio?
\endproblem
\enddocument

Figure 1. A TeX file (created by Acces).
Algebra 2A

Test on quadratic equations

Name ___________________________                Per/Sec. ______________

Show all work for credit. Good luck!


1.  \((1 + c)(13 + c) = 0\)                      2.  \(0 = p^2 - 6p - 55\)

3.  \(a^2 = 11a - 28\)                              4.  \(64 + 3h - 2h^2 = -h^2 - 44\)

5.  \(3d^3 + 33d^2 + 72d = 0\)                     6.  \(0 = a^4 - 26a^2 + 25\)

Part 2.    Solve for the indicated variable.

7.  \(s = \frac{rl - a}{r - l}; \text{ for } r\)     8.  \(ay^2 - y = 0; \text{ for } y\)

Part 3.

9.  The difference of two numbers is 6.          10.  A piece of wood is 16 in wide and 30 in long. How wide a strip must be cut off one end and one side to make the area of the wood 275 sq in?
    The sum of their squares is 116. Find the numbers.

11. A rectangular patio is surrounded on three sides by a fence (the remaining side is up against the house). If the area of the patio is 150 ft², and the total length of fence is 35 ft, what is the length and width of the patio?

Figure 2. A typeset document.
As you look over the TeX file, notice that the data (the actual contents) is distinct from the typesetting commands. For example, within a problem, there is no mention of line width, answer spaces, number or position. The ‘begin’ and ‘end’ commands reveal the structure of the document without imposing a particular style. In the publishing trade, this is known as generalized mark-up; it is a perfect use for TeX. (The program is less suitable for documents that are design-intensive, such as advertisements and brochures.) Fortunately, lessons and exams—and most other database applications—lend themselves to this approach. The proof is in the printed output.

The importance of generalized mark-up cannot be overstated. It is the reason Acces can print documents so many different ways. As you might guess, page layout can be changed simply by passing along a different value to `\textcolumns` or `\answerpos`, or by re-issuing the commands somewhere else in the document. Moreover, if you are skilled with TeX, you can change the meaning of `\beginproblem`, `\endproblem`, or any other command—in effect, re-program the typesetter. The main point is that you can create an entirely new page style without touching the data.

There are other advantages, too. For authors, this generalized form of problem-writing tends to be more efficient, because the focus is on content instead of design. And finally, for end-users, there is much greater flexibility. Since problems are formatted dynamically (when they are pulled out of the database, not when they are written), it is possible to re-arrange them and alter their appearance. The end-user can add a picture, insert a graph, delete an answer space, etc. If you have worked previously with a word processor or desktop publisher, then you know how difficult it is to revise or move around items that have already been formatted.

One question remains: how does TeX know what to do with an Acces-type document? Commands like `\beginproblem`...`\endproblem` must be defined somewhere, since they are not built into TeX. The answer lies in something called a format file. It contains all of the definitions, instructions and font information needed to process a document, and it is specially encoded for TeX. (The method for building a format file is beyond the scope of this manual.)

Some format files are general-purpose, others are designed for a specific task. Prof. Knuth wrote a format that is referred to as Plain TeX. He intended it for books, manuscripts, letters, etc., but many people find its low-level commands difficult to master. Another well-known format, with higher-level commands, is LaTeX, by Leslie Lamport. It is widely used for graduate theses, professional papers, and journal articles. Some organizations maintain their own format files, such as the American Mathematics Society, which insists that authors use its own AMSTeX when submitting papers.
Acces’ format file is called simply ACC, and is given the extension .FM1, .FM3 or .FMS, depending on your version of the program. The ACC format is something of a hybrid. It contains almost all of the typesetting commands found in Plain \TeX, but it is also modified for EducAide’s purposes; in other words, it is highly tuned for lessons and exams. Most of this guide is devoted to commands which are unique to ACC. (Appendix C points out some of the important differences between Acces’ format file and Plain \TeX.)

## Staying in control

Most of what you type is treated by \TeX the same as it is by a word processor or text editor. Numbers, letters and words are typeset just as you would expect. One of the main differences with \TeX is that it takes care of typographic details. This includes such things as line breaks, spacing, kerning and ligatures. About the only thing you must be on the lookout for are active characters (described below). Unless you want to give a typesetting command, change \TeX’s behavior, or produce some special effect, you can treat your keyboard like an old-fashioned typewriter.

Here are two of the most important rules when working with \TeX:

- Use normal keyboard characters. IBM PC’s and compatibles have an extended character set for line drawing, Greek letters, etc. But you should not use any of those characters, because they are non-standard and unrecognizable to \TeX. If you want to produce a special symbol, use the appropriate typesetting command. Similarly, low-order control characters like Tab (ASCII #9) and Form Feed (ASCII #12) should not be used. \TeX does not forbid them; however, their screen representations vary so much that they should be avoided.

- “Don’t force it.” Most of the time, you should not try to control \TeX. Instead, you should let the program work its magic. (This applies to mathematics above all.) There is a reason \TeX follows strict spacing rules and other conventions: output which is typographically correct looks better and is easier to read than that which is tinkered with excessively. Unfortunately, \TeX cannot provide the same kind of feedback as a word processor—you do not see immediately what your work will look like in print—but you will certainly appreciate its beauty and precision.

Afficionados will tell you there are three ways of controlling \TeX, or three sorts of typesetting commands. The names are not very important, but they are given here for the sake of completeness: control characters, control symbols, and control words. We will give a brief description of each, then drop the formalities and refer to them simply as commands.
**Control characters**

If you have read the Acces manual, you already know about control characters: they are single keyboard characters that are active. There are 10 altogether, though it is possible to define more: \ { } $ ^ _ \% & and #. A point made often is that such characters are not printed; instead, they control typesetting operations. The meaning of each control character is given in the chart below.

<table>
<thead>
<tr>
<th>control char.</th>
<th>meaning</th>
<th>control char.</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\</td>
<td>escape character</td>
<td>_</td>
<td>subscript</td>
</tr>
<tr>
<td>{</td>
<td>begin-group symbol</td>
<td>^</td>
<td>superscript</td>
</tr>
<tr>
<td>}</td>
<td>end-group symbol</td>
<td>~</td>
<td>tie words together</td>
</tr>
<tr>
<td>$</td>
<td>math on and off</td>
<td>&amp;</td>
<td>alignment character</td>
</tr>
<tr>
<td>%</td>
<td>comment character</td>
<td>#</td>
<td>macro parameter</td>
</tr>
</tbody>
</table>

Some of TeX’s control characters are commonplace. If you have programming experience or have tried sending commands to a printer, you may recognize the backslash as an escape character. In fact, the backslash is often used for that purpose. What it does is interrupt the normal flow of data—that is, it tells a device to get ready for some sort of control signal. The same is true with TeX. A backslash informs the program that the following character(s) are not to be typeset; rather, they are to be interpreted as a command.

Note: If you want to print an active character, the usual method is to put a backslash in front of it. For example, to print # or $, type \# or \$. The method works for all active characters except the backslash, tilde and hat. Though rarely necessary, you can print them by typing their names as control words—that is, type `\backslash`, `\tilde` and `\hat`. More information about control words is given below.

**Control symbols**

Control symbols are the second type of command. Two examples have already crept into this discussion: \# and \$. As you can tell, each one is a backslash followed by a symbol. (The formal definition is an escape character followed by a single non-letter, so numbers and punctuation marks qualify.)
The most common control symbols are listed in the table below. You can get by knowing just a few, namely those which print active characters. Unless you work with a foreign language, it is unlikely you will need to put accents on top of letters. The other control symbols perform even more specialized tasks, and may be looked up in a reference book if necessary (see Appendix D).

<table>
<thead>
<tr>
<th>control symbol</th>
<th>meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$</td>
<td>print a dollar sign</td>
</tr>
<tr>
<td>%</td>
<td>print a percent sign</td>
</tr>
<tr>
<td>&amp;</td>
<td>print an ampersand</td>
</tr>
<tr>
<td>#</td>
<td>print a pound symbol</td>
</tr>
<tr>
<td>_</td>
<td>print an underscore</td>
</tr>
<tr>
<td>&quot;</td>
<td>put an umlaut over the next letter</td>
</tr>
<tr>
<td>‘</td>
<td>put an acute accent over the next letter</td>
</tr>
<tr>
<td>‘‘</td>
<td>put a grave accent over the next letter</td>
</tr>
<tr>
<td>-</td>
<td>discretionary hyphen (insert if necessary)</td>
</tr>
<tr>
<td>*</td>
<td>insert a mult. sign and allow a line break</td>
</tr>
<tr>
<td>,</td>
<td>insert some extra space in a math formula</td>
</tr>
<tr>
<td>!</td>
<td>remove some space from a math formula</td>
</tr>
</tbody>
</table>

If the difference between control symbols and control characters is unclear, an example should help. Consider the two commands \$ and \$. Normally a dollar sign is a control character which means “turn math mode on or off.” When it is preceded by a backslash, it becomes a control symbol which means “print a dollar sign.” The control symbols \% \# and \& function in a similar manner.
Control words

The remaining commands, known as control words, are used most often. Briefly, a control word is a backslash followed by one or more letters. The case of letters does matter. The command \AbC, for instance, is different from \aBc. A large number of control words are built into \TeX{} (there are about 900 in its plain format), and most applications require even more (\Acces{} adds about 750). While control words include everything from \aleph{} to \zeta{}, the vast majority, which deal with page layout, may be safely ignored.

At first you might disagree with the wisdom of making upper- and lowercases distinct. But the advantages show up soon enough. Upper- and lowercase symbols, like Greek letters, are easier to produce. For example, in math mode, \delta{} yields δ and \Delta{} yields ∆. Also, uppercase letters can signify a more powerful command. Plain \TeX{} uses the control words \big{} and \Big{} to alter the size of parentheses, and you can easily guess which has the greater effect.

Here are two hints about typing control words:

- Use lowercase letters unless you have a good reason to do otherwise. No more than one or two percent of all commands use uppercase letters, and they make sense in only a few situations: (1) when a symbol has an upper and lower variant; (2) when a typesetting effect can be magnified; and (3) when a name is being spelled out (e.g., \Acces{} and \TeX{}). If you are in doubt, play it safe and use lowercase letters.

- Control words tend to be literal commands (the name of the thing you want to typeset). If a control word is not literal, then it is a simple abbreviation or mnemonic device. This is especially true of mathematical symbols and operators. If you are not yet convinced, here is another list of control words whose meanings you can guess: \angle, \approx, \infty, \iff, \pm.* As mentioned in the \Acces{} manual, you can do a lot with about two dozen commands, and they are easy to remember and quick to type.

*The answers: \&, \approx, \infty, \Theta, ±.
3 \textit{\TeX}nical issues

\section*{Spacing}

When it comes to spacing, \TeX{} handles almost everything for you. The program is smart enough to insert extra space after punctuation marks. It stretches and shrinks inter-word spaces so that lines are broken neatly, and it knows how to spread out the various parts of a math formula. But, alas, \TeX{} does not understand English and cannot know all of your intentions, so it sometimes needs a little help. This section covers the main rules about spacing and tells you a few things to try if you don’t like the rules. More options are given later.

Let’s begin with punctuation. Since it is dealt with automatically, you do not have to type two spaces after a period or any other punctuation mark. (Perhaps you are accustomed to doing that with your typewriter or word processor.) Multiple spaces do not upset \TeX{}; they are simply unnecessary. In fact, multiple spaces are always counted as one, so it does not matter how many of them you type.

Since multiple spaces are treated as one, you have quite a bit of flexibility as a problem-writer. For example, you can spread out the various parts of a math formula by pressing \texttt{spacebar} at any time. You can also use spaces to represent various kinds of alignments, such as systems of equations and matrices. In fact, we encourage you to do this, especially if you will be sharing your data with others. (Raw data is much easier to read if there are plenty of spaces.) Just don’t forget that the extra spaces are for your benefit; they do not change the way a problem is typeset. To line things up or move things around, you must still use the appropriate commands.

\TeX{} also overlooks all spaces at the beginning of a line, and it changes carriage returns (end-of-line markers) into spaces. There are several reasons for doing this. First, \TeX{} tries to accommodate programmers, who are fond of indenting lines. Their indents, whether formed by spaces or tabs, do not affect anything. Second, it discourages people from trying to change margins or paragraph shapes by pressing \texttt{spacebar}. Those things are governed only by typesetting commands. Of course, when typing, you may want to represent an indented paragraph, and spaces may be conveniently used for that purpose.

What about the end of lines? You already know that multiple spaces are counted as one. A bigger concern is the absence of spaces. Word processors usually strip away blanks from the end of a line. Since lines on the screen do not necessarily match lines on the printed page, it is necessary for \TeX{} to restore those blanks, by converting carriage returns into spaces. Also, there is no visible representation of a space at the end of a line, so \TeX{}’s method assures that at least one space is present. (It makes no difference if there is more than one.)
Finally, \TeX{} ignores all spaces that come after a control word. For example, if you type \TeX{} nicality, you will see the word \TeX{}nicality in print. This curious behavior stems from the definition of a control word: an escape character followed by one or more letters. \TeX{} needs to know when a control word has ended, and a space is one of the best indicators. You can also signal the end of a control word by typing a number, symbol or other command.

If you really want a space to appear in print, regardless of the circumstance, \TeX{} provides an easy solution: a control-space. In most books about \TeX{}, the command is represented as \\_. That fancy representation just means type a backslash followed by a space. The symbol _ is a reminder to include the space; it does not get printed. The technique is a useful one to remember. It enables you to insert a space after a control word or in a math formula.

To summarize, here are \TeX{}’s rules about spacing:

- Multiple spaces are counted as one.
- A carriage return is treated like a space.
- Spaces at the beginning of a line are ignored.
- Spaces after a control word are ignored.
- A control-space is always printed as a normal space.

### Grouping & Parameters

Commands perform a wide variety of tasks—everything from setting up the length of a page to hyphenating a word. Commands also behave differently from one another. Some of them involve parameters, which means they expect input from you. Others act like switches, which means they turn on and off typesetting operations. As it happens, the simplest commands are used most often: those which print letters and symbols.

The present discussion focuses on the effect of commands, how to limit their scope and, in some cases, how to broaden it. A good subject to start with is fonts. The principles apply to many other areas of operation.

Nearly every version of \TeX{} provides these three font commands: \texttt{\rm{}}, \texttt{\it{}} and \texttt{\bf{}}. The first produces a roman font, the style of print you see here. The second produces an italicized version of the font and the third, a bold version (the command is an abbreviation for “bold face”).
Whenever you type something—a problem, memo or set of directions—the command \texttt{\textbackslash rm} is implied at the beginning. The roman font remains in effect until you issue a different command. So the question arises, how do you change fonts temporarily? Two methods are commonplace with \TeX.

The first method is to treat font commands like switches. In this case, you type \texttt{\textbackslash bf} or \texttt{\textbackslash it} and the text you want to emphasize. Then, to restore the original font, you type \texttt{\textbackslash rm}. The basic idea is to activate or turn on one font after another. To illustrate,

\begin{verbatim}
\bf bold words \rm and \it italic words
\end{verbatim}

yields \textbf{bold words} and \textit{italic words}.

The second method takes advantage of \TeX’s grouping mechanism. By now you probably know that groups are formed by curly braces. One of their main purposes is to limit the effect of a command. To produce a single bold word, like this, you type the command \texttt{\textbackslash bf} and the word inside a group, like \texttt{\{\textbackslash bf this\}}. Of course, a group may contain more than a single word. The result of

\begin{verbatim}
\{\bf bold words\} and \{\it italic words\}
\end{verbatim}

is the same as the example in the last paragraph. The important point is that everything is restored to normal at the end of the group.*

Note: There are implicit braces around each item in \textit{Acces}' documents. In other words, each problem, direction and memo is put inside a group. This keeps all font changes and other commands local. It also confines typesetting errors to a small amount of text.

\textit{Acces} provides another method for changing fonts, namely the commands \texttt{\textbackslash italic} and \texttt{\textbackslash bold}. The commands are not part of the plain \TeX\ format. Nor are they synonyms for \texttt{\it} and \texttt{\bf} (their results are similar but their usage is different). The only reason for presenting \texttt{\textbackslash italic} and \texttt{\textbackslash bold} now is to illustrate a concept related to grouping—that of \textit{parameters}.

Some commands are defined in a way that they operate on a limited amount of text. \texttt{\textbackslash italic} and \texttt{\textbackslash bold} are good examples. Each command has one \textit{parameter}. In simple terms, this means you must provide some input or type something after the command. What you type is called an \textit{argument}. An argument is usually a number, dimension (like 2cm) or, in the case of \texttt{\textbackslash italic} and \texttt{\textbackslash bold}, some general text.

*Actually, things are restored to their previous state. If you type, for example, \texttt{\bf some text \{\textit{more text}\}...} the command \texttt{\bf} will be in effect after the group.
The way TeX handles arguments is interesting and turns out to be very convenient. All commands which are described in this manual—and which take arguments—behave in the same manner: they treat the next character that you type as their input unless you indicate differently. In other words, an argument is by default a single digit, letter, symbol or command. This makes it very easy to produce something like $\sqrt{5}$. You start with the command $\sqrt$, which produces a radical, and you add one argument, the radicand. In math mode, you type simply $\sqrt{5}$.

How do you indicate that an argument is more than a single character? The answer should come as no surprise: put the argument inside a group—i.e., between a pair of curly braces. The technique was mentioned in the Access manual, in regard to sub- and superscripts. Grouping symbols are commonly used to delimit an argument or, in effect, broaden the scope of a command. Returning to the subject of fonts, this is how you would produce bold words and italic words, using the method just described:

\begin{verbatim}
\textbf{bold words} and \emph{italic words}.
\end{verbatim}

Notice that the arguments, not the commands themselves, are inside groups.

Another important question about parameters is this: How do you know which commands have them? Most books about TeX follow a certain convention. If a command has a parameter, then what you are supposed to type is shown inside angle brackets, something like \texttt{\textbackslash command\{argument\}}. Sometimes the argument is given a name, but only when the name is commonplace—e.g., \texttt{\textbackslash seg\{endpoints\}}. Otherwise, there is a brief description of what you are supposed to type. For example, \texttt{\textbackslash ital\{text material\}} means that the command \texttt{\textbackslash ital} operates on ordinary text.

A few commands have more than one parameter. A good example of this is a fraction, formed by the command \texttt{\textbackslash f}. As you might expect, the command has two parameters. Its representation is \texttt{\textbackslash f\{numerator\}\{denominator\}}. For simple fractions, like one-half, you can conveniently type \texttt{\textbackslash f12}. For more complicated fractions, you may need to use grouping symbols around the numerator or denominator or both. The fractions

$$\frac{3}{10}, \frac{25}{2} \text{ and } \frac{6a}{8a^2}$$

are obtained by typing \texttt{\textbackslash f3{10}}, \texttt{\textbackslash f{25}2} and \texttt{\textbackslash f{6a}{8a^2}} in math mode.

Let us conclude this section with a reminder about control words. A special situation arises when a control word takes an argument, and the argument happens to be a single letter. In that case, you need to distinguish the control word and the argument. You may either put a space between them or put the argument inside a group. To illustrate, you can produce $\sqrt{x}$ by typing $\sqrt \ x$ or $\sqrt \{x\}$. But you cannot type $\sqrt \{x\}$. Do you see why? \TeX would assume you were issuing a command made up of the letters ‘rx’ (the command does not exist and would be ignored).
Note: If you don’t want to be bothered with a lot of technicalities, you can make it a practice of putting every argument inside a group. In fact, this is the best thing to do if you want to automate your problem-writing with macros or recorded scripts. You may use all of the commands in this manual with the knowledge that extra grouping never hurts anything, as long as left and right braces are matched up properly.

## Paragraphs

One of the most important differences between \TeX{} and ordinary word processors (or desktop publishers) is the handling of paragraphs. Word processors have a feature called “word wrap,” which means they begin a new line as soon as the current one is filled up. \TeX{}, on the other hand, waits until the end of a paragraph before breaking any lines. This enables it to locate optimal breakpoints instead of merely convenient ones. The feature is extremely powerful and is one reason professional typesetters have such high regard for the program.

\TeX{}’s line-breaking mechanism has particular advantages for Acces. First, it offers a lot of flexibility when writing problems, because you do not need to be concerned about the length of lines. In fact, the length of lines in your data files is completely irrelevant to \TeX{}, as are carriage returns (end-of-line markers). For example, it makes no difference whether you type

\begin{verbatim}
The cow jumped over the moon
\end{verbatim}

or

\begin{verbatim}
The cow
jumped
over
the moon.
\end{verbatim}

\TeX{} will typeset both sentences exactly the same way.

Inside your word processor, you can press Enter whenever it is convenient to do so. When reading your data, \TeX{} treats carriage returns as ordinary spaces. Not until the typesetting stage—i.e., the formation of paragraphs—are lines really broken. This turns out to be an ideal way of handling a database. Instead of problems having a predetermined width, their dimensions can vary, depending on the number of columns, the presence of answer spaces, etc.
There are two ways of signaling \TeX that a paragraph has ended. The first way is to type a blank line (press \texttt{Enter} twice in a row). The other is to type the command \texttt{\par}. We recommend the second method, since it keeps the number of input lines to a minimum. You may, however, prefer the visual clue that a blank line provides—the choice is entirely yours.

Important: If an item contains only one paragraph, you do not have to do anything special. The command \texttt{\par} is implied at the end of each problem, direction and memo.

As you can tell, the formation of paragraphs does not require much extra work on your part. There are, however, two restrictions. The first is that you are not allowed to end a paragraph while in math mode (you wouldn’t want to do this, anyway). The second is that you cannot end a paragraph when typing an argument—that is, the input to a command.* For example, the command \texttt{\ital} takes one argument, but you should never type

\texttt{\ital{Some text...\par Some more text...\par}}.

The reason for these restrictions is that they help \TeX catch errors, such as unmatched braces and dollar signs. And by the way, the restrictions apply equally to \texttt{\par} and blank lines.

As for the shape of paragraphs, they are neither indented nor right-justified. This is a characteristic of the \texttt{ACC} format, not \TeX in general. Experience shows that numbered items look better this way. \texttt{Access} does insert a small amount of space between paragraphs in order to distinguish them. If you want to insert more space or indent the first line, you will have to use commands like \texttt{\smallskip} and \texttt{\indent}, which are described in the next chapter.

---

*This is a slight exaggeration. Some commands allow \texttt{\par} in their arguments. Nevertheless, you should assume it is not allowed unless you are told otherwise.
Text vs. Math

Perhaps you are already familiar with \TeX's way of doing math. The idea is simple: you put all math formulas between a pair of dollar signs. When \TeX encounters the first dollar sign, it goes into “math mode”; when it reaches the second dollar sign, it returns to normal typesetting or “text mode.” This allows you to mix math and text freely, since you can insert formulas in the middle of sentences, between paragraphs, anywhere you like.

Why should there be a separate mode for math? The main reason is that the spacing of formulas is very different from ordinary text, and there are symbols and operators which require special treatment. Since \TeX cannot know what is math material, dollar signs are used to signal the program. Turning on and off math mode may seem like extra work, but it actually makes your job easier. When typesetting formulas, \TeX treats letters as variables (it italicizes them), it turns hyphens into true minus signs, it adjusts the size of sub- and superscripts, and much more. The great news is that it does these things automatically.

Many additional commands become available to you when typing math formulas. In fact, a good portion of \TeX’s commands may be used in math mode only. That is true of mathematical constructs like fractions and radicals, as well as special symbols: operators, relations, delimiters, etc. Also, a few commands behave differently in math mode. For example, in text mode, the commands \d and \t are used for ‘dot’ and ‘tie’ accents, as in \textcircled{a} and \textcircled{o}. In math mode, they produce division and times symbols (\div and \cdot).

The next chapter describes about 150 commands for problem-writing. Those which must be used in math mode are grouped together, to remind you of their special status. Of course, it is inevitable that you will use a command in the wrong mode, or you will turn math on and forget to turn it off. That is not a disastrous thing—sometimes you will even get away with it. \TeX can identify many errors and correct them for you. In particular, it turns math on and off if it encounters a command that is used in the wrong mode.

Error-checking usually results in math mode being turned on, because the signs are so obvious (e.g., a fraction or exponent suddenly appears). It is more difficult for \TeX to know when a formula has ended, and when to turn math mode off. Thus, as a precautionary measure, \TeX does not allow a formula to continue beyond a single paragraph. The program always quits math mode when it comes across a \texttt{\par} or related command. (As a separate measure, \texttt{Acces} restores everything to normal at the end of a problem. It is almost impossible for any error to spill over from one problem to the next.)
How do you know if you’ve forgotten to turn math on or off? The error is typically manifested in a long string of italicized letters. For example, if you leave out the dollar signs when typing

\[ e=mc^2 \text{ is well-known} \]

you will see in print:

\[ e=mc^2islwell \text{ – known}. \]

The exponent causes \TeX{} to turn on math mode. But the program has no way of knowing where the formula ends, so it changes subsequent letters into variables. Unfortunately, the only way to correct the problem is to go back and insert dollar signs in the appropriate places.

Here are the main ideas about math mode:

- Any formulas which are inherently mathematical should be typed between a pair of dollar signs.
- \TeX{} handles all spacing in math mode.
- Any spaces that you type are ignored.
- Letters are treated as variables (they are typeset in an italicized math font).
- Some commands operate in math mode only. If necessary, \TeX{} will turn math on when it encounters one of those commands.
- \TeX{} does not always know when to turn math off, so you should be sure that dollar signs are matched up correctly.

## Ties & Comments

Some of \TeX{}’s commands have no obvious effect in printed documents. But you, the author, still need to use the commands. A case in point is the control character \~{}, which ties words together. Its effect is important though subtle: a tie shows up as an ordinary space, but it prevents a bad line break from occurring. One instance where a line break is considered bad is between a name and title, as illustrated by Dr. Welby. (Do you see what just happened? That distracting break could have been avoided by typing Dr.~Welby.) The next chapter will tell you about other situations where a tie comes in handy.
Why should you pay attention to such details? The main reason is that you do not know ahead of time how problems will be typeset. In particular, you do not know what font size and page layout will be chosen inside of **Acces**. Both choices affect line breaks. That is why problem-writing requires some anticipation of worst cases. Of course, you may get away with short-cuts. But a little extra effort is all it takes for your problems to be typeset reliably and professionally.

The percent sign is another interesting command. Its purpose is to *keep things out* of a printed document. As you may have discovered already, **Tex** ignores everything on a line that comes after a `%`. That is why it is called a comment character. It lets you insert comments into a file or attach notes to a problem, without the notes being printed. If you find that a problem ends abruptly, or part of a line is missing, it is probably because you have used a percent sign incorrectly.

There are two good reasons for putting comments in your files. First, they are reminders to yourself. Since your objective is to store problems in a database, perhaps for a long time, you may need a little refresher later on. For example, you may want to know how you arrived at an answer or why a problem was written in a certain way. Second, comments are helpful to those who share your data. Even if you think no one else will ever see your files, there may come a day when someone *will* ask for them and wish you had explained things better. (Any computer programmer will tell you that file comments are vitally important.)

Note: The `%` sign can serve another purpose: it can block out a carriage return at the end of a line. This is a pretty sophisticated technique; it is relevant mainly to advanced users. Recall that **Tex**’s normal procedure is to turn a carriage return into a blank space. If you do not want that to happen, type a `%` as the last character on the line (before pressing Enter).
4 Summary of Commands

This chapter describes many of the typesetting commands that are available. It is not a complete reference to \TeX because the program is well-documented in other places. Also, as mentioned in the introduction, the focus is on commands which are unique to Acces. Nevertheless, you will find enough information to write most kinds of problems, and there are plenty of hints and examples, so you will have no trouble using the commands for your own purposes.

We recommend that you familiarize yourself with \TeX before reading any further. If you have not used the typesetter previously, you should definitely skim over the previous two chapters. In addition, you may find it helpful to look at the sample files that come with the ‘Extra’ database. You can get a good idea of how commands work by looking at actual problems. Then you can turn to this chapter to learn the details.

Important: The ACC format file is revised periodically. If a command does not work as it should—or rather, as it is described—your format file may not be up-to-date. In that case, you can obtain a more recent version from EducAide.

In this chapter, commands are not listed alphabetically but are grouped together by topic. There are five main sections (the last one is for advanced users):

- Ordinary text
- Common symbols
- Math formulas
- Special effects
- Document layout

Within each section, related commands are presented all at once, or one after the other. If you want to look up a specific command, you can use the index at the back of this manual or the quick-reference charts in Appendix A.

Each command is marked with a \TeX or \EA. The first mark indicates that the command is part of Plain \TeX, the standard implementation of the program. The second indicates that the command was defined by EducAide and is part of the ACC format. For all practical purposes, it makes no difference where the command comes from, although you can look up plain commands elsewhere.

Finally, keep in mind there are many techniques for problem-writing, and there are hundreds of additional \TeX commands that are beyond the scope of this manual. If you have particular needs that are not addressed, please contact EducAide or refer to Appendix D, which lists other resources.
% (comment character)

This command is used to “comment out” notes or reminders to yourself. You can use it as often as you like when typing text or math material. When reading a line, TeX ignores everything that comes after the % (up to the end of the line). In other words, what you type after the % is part of your raw data but is never printed.

Note: If you want to print a percent sign, instead of using it as a comment character, put a backslash in front of it. This technique is shown in the example.

Example

Question: % Do you know how to print a percent sign?
% Ignore the previous question. Here’s the real one:
What is 125\% of 144?

results in...

Question: What is 125\% of 144?

~ (tie)

A tie inserts a normal inter-word space and prevents a line break from occurring. Though never required, the command is useful as a precautionary measure, since you do not know ahead of time where line breaks will occur. It also helps TeX make intelligent decisions, since the program does not understand rules of grammar. The example below shows several situations where the command is appropriate.

Note: In the example, a line break would obviously not occur after the first word. But another principle is illustrated: a tie forces TeX to use a normal inter-word space. Without the tie, the program would assume that “Mr.” is the end of a sentence, and would put more space after the period.

Example

Mr.~Jones has twice as many coins as his son, who has 4~more than Mrs.~Jones. How many coins does each person have, if the total number is~52?

results in...

Mr. Jones has twice as many coins as his son, who has 4 more than Mrs. Jones. How many coins does each person have, if the total number is 52?
\TeX \ \texttt{par}

\TeX gives you two ways of ending a paragraph: insert a blank line (press \texttt{Enter} twice in a row) or type the command \texttt{par}. There are advantages to each. The first method is traditionally used by typists as a way of separating paragraphs. The second lets you pack several paragraphs together, so that they appear (in raw form) as a single item.

The paragraph command is important because it activates \TeX’s line-breaking mechanism. The program waits for the end of a paragraph to figure out break points. For that reason, you should never type \texttt{par} (or insert a blank line) while math mode is on. Use the command after sentences or completed formulas only.

Note: In the ACC format, a little extra space is inserted between paragraphs. If you do not want the space, consider using the command \texttt{newline} instead of \texttt{par}. That may be appropriate when a paragraph is very short (a few words or some math material). Also, if an item contains just one paragraph, you do not need to type \texttt{par}; the command is implied at the end of every problem, direction and memo.

► Example

Graph each equation on the same coordinate system. (Be sure to label important points.)

\[ x = y \par y = -5 \par 3y = -12x \par y = 4x + 2 \]

results in…

Graph each equation on the same coordinate system. (Be sure to label important points.)

\[ x = y \]
\[ y = -5 \]
\[ 3y = -12x \]
\[ y = 4x + 2 \]
Normally you can start a paragraph without doing anything special. If you type any letter or number, or turn on math mode, \TeX will assume you are starting a new paragraph. In a very few cases, you need to be explicit and use the command \noindent or \indent. As you can guess, \indent adds space to the beginning of the first line, while \noindent makes everything flush left.

By default, Acces begins new paragraphs without any indentation. In other words, \noindent is implied at the beginning of every paragraph. Numbered exercises usually look better this way. If you disagree, you can override the default by giving the command \indent. \TeX allows the combination \noindent\indent. In fact, you can type \indent repeatedly, or use it to insert extra space anywhere in a paragraph, not just at the beginning of the first line.

\textbf{Example}

This is a typical first line of a paragraph, with some extra text, so you can see where the line break occurs.\par \indent This is an indented second paragraph.\par \indent\indent The third paragraph is indented even more.

results in…

This is a typical first line of a paragraph, with some extra text, so you can see where the line break occurs.

This is an indented second paragraph.

The third paragraph is indented even more.

This command is an abbreviation for “bulleted item”; it produces a round bullet in front of an indented paragraph. This particular style has become popular in recent years. Be careful not to overuse it. Bullets may be appropriate at times, but they tend to be distracting on tests and worksheets, especially when there is a lot of mathematical material.

Note: \bulit contains an implicit \par command, so it ends the previous paragraph before starting a new one. You can put blank lines around bulleted items, to help distinguish them (in raw form), but this is not required. Also, notice the use of \RT in the example; this puts a little extra space after the name of the postulate.
Example

These postulates come from a traditional Euclidean geometry course. Use them to prove that two intersecting lines share exactly one point.

- **POINTS POSTULATE** A line contains at least two points; a plane contains at least three noncollinear points; space contains a least four noncoplanar points.
- **LINE POSTULATE** Two points are in exactly one line.

results in...

These postulates come from a traditional Euclidean geometry course. Use them to prove that two intersecting lines share exactly one point.

- **POINTS POSTULATE** A line contains at least two points; a plane contains at least three noncollinear points; space contains a least four noncoplanar points.
- **LINE POSTULATE** Two points are in exactly one line.

\begin{Verbatim}
\smallskip \medskip \bigskip
\end{Verbatim}

Instead of being indented, paragraphs are separated by a small amount of space. If you want to insert more space between paragraphs, type \texttt{\smallskip}, \texttt{\medskip} or \texttt{\bigskip}. The effect of each command varies with font size. This is how much space is inserted, given a font size of 10 points:

\begin{Verbatim}
\smallskip \rightarrow \quad \medskip \rightarrow \quad \bigskip \rightarrow \quad
\end{Verbatim}

In short, \texttt{\medskip} produces about the same amount of space that you normally see between paragraphs, while \texttt{\smallskip} produces half as much and \texttt{\bigskip} twice as much.

You may create additional space by typing one of the commands repeatedly, or by combining them—e.g., \texttt{\bigskip}\texttt{\medskip}. Keep in mind, however, that these commands add to the space already present between paragraphs; they do not substitute for it.
\texttt{\texttt{\textbackslash newline}}

This command forces a line break. Typing \texttt{\textbackslash newline} is like pressing \texttt{Enter} inside a word processor, with one crucial difference: \TeX{} obeys the command while it ignores end-of-line markers (carriage-returns).

The command may be used within a paragraph or between paragraphs. It may also be typed repeatedly—e.g., \texttt{\textbackslash newline\textbackslash newline}—to produce more than one blank line. If you would like to insert just a little extra space between two lines of a paragraph, \texttt{\textbackslash newline} may be used in conjunction with \texttt{\textbackslash dn} (see below).

Note: In the example, \texttt{\textbackslash newline} is combined with \texttt{\textbackslash indent} to distinguish one word from the next. There are other ways of writing this sort of problem; see especially the commands \texttt{\textbackslash par} and \texttt{\textbackslash bulit} (above) and \texttt{\textbackslash subpart} (in the section called \textit{special effects}).

\paragraph*{Example}

Sketch these objects on the back of your test booklet, in the space provided (be sure to label each one):
\texttt{\textbackslash newline\textbackslash indent trapezoid}
\texttt{\textbackslash newline\textbackslash indent concave polygon}
\texttt{\textbackslash newline\textbackslash indent regular hexagon}

\textit{results in}...

Sketch these objects on the back of your test booklet, in the space provided (be sure to label each one):
\begin{itemize}
  \item trapezoid
  \item concave polygon
  \item regular hexagon
\end{itemize}

\texttt{\texttt{\textbackslash dn \textbackslash Dn \textbackslash DN}}

These commands insert a little extra space between the \textit{current} and \textit{next} line of a paragraph. The amount of space varies with font size. This is how much space is inserted, given a font size of 10 points:
\begin{align*}
\texttt{\textbackslash dn} & \rightarrow \quad \texttt{\textbackslash Dn} & \rightarrow \quad \texttt{\textbackslash DN} & \rightarrow \\
\end{align*}

The commands are similar to \texttt{\textbackslash smallskip}, \texttt{\textbackslash medskip} and \texttt{\textbackslash bigskip}, except for one thing: \texttt{\textbackslash dn}, \texttt{\textbackslash Dn}, and \texttt{\textbackslash DN} are used \textit{within} a paragraph, not \textit{between} paragraphs. For that reason, the commands are often combined with \texttt{\textbackslash newline}. 

\textit{Summary of Commands}
Note: \dn is used in the second example because the fraction is rather tall; although it is not absolutely necessary, the extra space is a nice touch. Also note that the command has a visible effect only because the paragraph is narrow. In other situations, it is possible that everything would fit on a single line and there would be no need for the extra space. In that sense, the command \dn is a precautionary measure, like the ~ character (tie).

\textbf{Example}

Which points are on the line $y=4x-3$?
\newline $A(4,3) \ \text{Rt} \ B(1,1) \ \text{Rt} \ C(-1,1) \ \text{Rt} \ D(0,-3)$

Give the excluded values of $x$ (if any) in the following expression: \newline $\frac{2x^3-x^2-x}{x^2+3x-10}$.

\textit{results in...}

Which points are on the line $y = 4x - 3$?
$A(4,3)B(1,1)C(-1,1)D(0,-3)$
Give the excluded values of $x$ (if any) in the following expression: $\frac{2x^3-x^2-x}{x^2+3x-10}$.

\textbf{\LaTeX} \textit{-- and ---}

In text mode, two or three successive hyphens produce a dash. Two hyphens produce an “en-dash”, which is used in number ranges. Three hyphens produce an “em-dash”, which is used as punctuation in sentences. Do not confuse dashes with a negative sign, which occurs only in math mode, or a single hyphen, which is used to break up a word.

\textbf{Example}

You can find en- and em-dashes in various places—for instance, look at chapters 1--3.

\textit{results in...}

You can find en- and em-dashes in various places—for instance, look at chapters 1–3.
Two single quotes, when typed in succession, become a double quote. In other words, when you type ‘‘, you see “ in print. The keyboard character "char34 may also be used for quotations, but it does not look as nice (in fact, some implementations of \TeX\ change its meaning, so that it is never used accidentally).

Note: In text mode, the keyboard character ’ produces an apostrophe or single right quote. In math mode, it gets turned into a prime symbol.

\textbf{Example}

‘Real’ opening and closing marks are superior to old-fashioned typewriter characters. A professional typesetter will say, ‘‘Don’t put ” and ” around any quotation of mine!’’

\textit{results in…}

‘Real’ opening and closing marks are superior to old-fashioned typewriter characters. A professional typesetter will say, “Don’t put ’ and ’ around any quotation of mine!”

\textbf{\TeX} \texttt{\textbackslash \textit{it \textbf{bf}}}

These commands change fonts: \texttt{\textbackslash \textit{rm}} produces the font you see here; \texttt{\textit{it}} produces an italicized version of the font; and \texttt{\textbf{bf}} produces a bold version. The three fonts are part of a typeface collection that \TeX\ calls “Computer Modern,” and which \textit{Access} refers to simply as “Roman.”

If you want to limit the effect of these commands, you should use them inside a group—i.e., between a pair of curly braces. Otherwise, a font command will continue to operate until a different command is given. (In no event does a font command continue beyond a single problem, direction, or memo.)

The font commands operate in math as well as text mode. In math mode, they affect the typesetting of variables (letters) only. If you give the command \texttt{\textbackslash \textit{rm}}, for instance, \TeX\ will use the upright roman font instead of the italicized math font. That may be preferable when you are dealing with points or dimensions.

Note: \texttt{\textbackslash \textit{rm}} is implied at the beginning of each problem that you type. You do not need to use the command unless you switch to another font and then want to switch back to roman. Also, in the second example, no grouping symbols are used with \texttt{\textit{it}}, because the command operates until the end of the problem (when everything is restored to normal).
► Example

Avoid using a lot of `{\bf bold words}` and `{\it italicized words}`, especially in the same sentence. They can be `{\it very}` distracting.

Define these types of angles: `{\it adjacent, vertical, supplementary}`.

results in...

Avoid using a lot of **bold words** and *italicized words*, especially in the same sentence. They can be *very* distracting.

Define these types of angles: *adjacent, vertical, supplementary*.

\begin{itemize}
  \item \texttt{\textit{ital \bf bold}}
\end{itemize}

These commands, which produce italicized and bold text, are alternatives to `{\it}` and `{\bf}`. Instead of operating like switches, each of these commands has one parameter. For example, to get italics, you type `{\it}{\textit{text material}}`.

Why use `{\it}` instead of `{\it}` (or `{\bf}` instead of `{\bf}`)? While it is true that the commands have the same effect, they operate very differently. The command `{\it}` is most often used inside a group, so as to limit its effect. With `{\it}`, the text material—not the command—is put inside a group. So which approach is better? The answer really comes down to personal preference.

► Example

For extra emphasis, use `{\it}\textit{italics}`. For even more emphasis, use `{\bf}\textbf{bold letters}`.

results in...

For extra emphasis, use *italics*. For even more emphasis, use **bold letters**.
This is a general purpose command for underlining text. It has one parameter, as in \undl(\text{text material}). While the command is similar to \ital and \bold, it is not a font effect \textit{per se}, since it does not change the typesetting of any characters. Instead, it puts a line under a block of text.

You should not use the command \undl on more than a few words at a time. If you try to underline too much text, some of it will stick out into the margins. The reason is that the text forms an unbreakable box; it is not compatible with \TeX’s line-breaking mechanism.

\undl can be used in combination with other font commands. As shown in the example, you can produce bold underlined text. This nesting of commands works in either direction—e.g., you can type \undl{\bf...} or \bold{\undl{...}}.

\textbf{Example}

\undl{\bf Extra Credit}\par
Solve the previous problem again, this time over the set of \undl{complex numbers}.

results in...

\textbf{Extra Credit}
Solve the previous problem again, this time over the set of complex numbers.

\textbackslash char
The command \textbackslash char forces \TeX to insert a normal inter-word space. Its main purpose has do with control words. As you recall, a control word is a backslash followed by one or more letters, and \TeX ignores any spaces that come after it. When you type the command \textbackslash char, you are simply telling \TeX not to ignore the space.

Note: The symbol \textbackslash char is a reminder that you are supposed to press spacebar; the symbol does not appear in print. By the way, at the end of a typewritten line, you may press Enter instead of spacebar. An end-of-line marker (carriage return) has the same effect as a space.
Example

Compare these statements:
\TeX is a computerized typesetter.
\TeX is a computerized typesetter.

results in...

Compare these statements:
\TeX is a computerized typesetter.
\TeX is a computerized typesetter.

\[\begin{array}{c}
\text{rt} \\
\text{Rt} \\
\text{RT}
\end{array}\]

These commands increase the amount of space between words and symbols. The amount of space varies with font size and it is allowed to stretch or shrink by about 15%. (This may seem trivial, but it helps \TeX produce nice-looking paragraphs.) This is how the commands relate to a normal inter-word space, given a font size of 10 points:

normal $\rightarrow$ $|$ $\rt$ $\rightarrow$ $|$ $\Rt$ $\rightarrow$ $|$ $\RT$ $\rightarrow$ $|

It does not matter whether the commands follow a space. For consistency, they absorb any space that precedes them (including each other). Thus, you should not bother combining them or typing one of them repeatedly. If you want to insert more space than the command $\RT$ provides, you can use other commands such as $\qquad$ (see below).

Note: $\rt$ produces the same amount of space that you normally see after a period or colon; its main purpose is to separate text and equations. $\Rt$ and $\RT$ have a more pronounced effect; they are good for highlighting replacement values, among other things.

Example

Solve: $3x>-12$ $\rt$ and $\rt-x+8<=6$ $\par$
Evaluate $2a-(3-a)$, $\Rt$ for $a=-5$

results in...

Solve: $3x > -12$ and $-x + 8 \leq 6$
Evaluate $2a - (3 - a)$, for $a = -5$
These commands insert one and two \textit{ems} of space. An em is just a fancy way of referring to the width of the letter ‘m’. Like other spacing commands, the effect of \texttt{\quad} and \texttt{\qquad} varies with font size. Here is the amount of space they insert, given a font size of 10 points:

\texttt{\quad} \rightarrow | | \texttt{\quad} \rightarrow | |

In actual problem-writing, you are unlikely to need the commands \texttt{\quad} and \texttt{\qquad}. They are described here for two reasons. First, they are part of nearly every implementation of \TeX and you would probably run into them anyway. Second, the command \texttt{\qquad} is convenient for illustrating things. It separates math formulas by just the right amount of space, so that you can see how they are typeset (it also makes composite statements easier to read). \texttt{\qquad} is used in many examples in this chapter.

\textbf{Example}

Therefore, for $x<>0$, \par\indent
\texttt{$F=x*f \quad \text{and} \quad F=-x*f^{-1}$} \par

results in…

Therefore, for $x \neq 0$,

\begin{align*}
F &= x \times f \\
F &= -x \times f^{-1}
\end{align*}

\textbf{EA} \texttt{\ \ (thinspace)}

The command \texttt{\ \} performs two jobs: it inserts a thinspace between words and symbols, and in math mode it changes the italicized font into an upright roman font. The first job may be performed by the Plain \TeX command \texttt{\thinspace}, and the second by the command \texttt{\rm}. So why are those functions combined into a double-backslash?

The answer lies in the most important use for the command, to separate numbers and abbreviated dimensions—cm, km, ft, mi, etc. When dimensions are abbreviated, they should not follow an inter-word space. Nor should they be italicized, which normally happens in math mode. Since dimensions are commonplace in math problems, the double-backslash turns out to be very convenient.
Like other spacing commands, the effect of $\backslash\backslash$ varies with font size. Given a 10 point font, it inserts this much space →$\|$. Unlike other spacing commands, however, a $\backslash\backslash$ joins together what comes before and after it. In other words, a line break will not occur at a thinspace.

Note: The example demonstrates both a tie and double-backslash. A tie is more suitable for dimensions that are not abbreviated. The ~ character prevents a line break from occurring and inserts a normal inter-word space, but it does not automatically change fonts in math mode.

**Example**

If a woman runs 12$\mathrm{mph}$, how far can she go in 7~minutes?

The area of a rectangular carpet is 42$\mathrm{sq\,ft}$. If the length of the carpet is 5.6~feet, what is the width?

Find the momentum of a train whose speed is 14$\mathrm{m/s}$ and mass is $4 \times 10^5 \mathrm{\,kg}$.

*results in...*

If a woman runs 12~mph, how far can she go in 7 minutes?

The area of a rectangular carpet is 42 sq ft. If the length of the carpet is 5.6 feet, what is the width?

Find the momentum of a train whose speed is 14 m/s and mass is $4 \times 10^5$ kg.
Common symbols

All of the commands in this section can be used in text or math mode. This is an important point because some format files, such as Plain TeX, do not allow any math symbols to be used in text mode. The restriction is supposed to keep you from mistyping formulas, but it often does more harm than good. With the ACC format, ordinary symbols can be used anywhere. By ordinary, we mean such things as Greek letters, the infinity sign (∞) and script ℓ. The term does not include relations, operators and delimiters; these must be used in math mode (see next section).

\begin{itemize}
\item \texttt{\textbackslash rn \textbackslash es}
\end{itemize}

These commands produce symbols for the set of real numbers and the empty set—that is, \( \mathbb{R} \) and \( \emptyset \). For real numbers, you could also use the Plain TeX command \texttt{\textbackslash Re}, which produces the fancier symbol \( \mathbb{R} \).

Note: In the example, a control-space is used after \texttt{\textbackslash es} to produce a normal inter-word space, and a tie is used before \texttt{\textbackslash rn} to prevent a distracting line break. Also, notice that math mode is turned on for all equations (formulas); this is necessary for proper spacing.

\begin{itemize}
\item \textbf{Example}
\end{itemize}

If you are asked to solve these equations, \( x = x + 1 \) and \( x + (-x) = 0 \), your answers should be \texttt{\textbackslash es} and \texttt{\textbackslash rn}.

\begin{itemize}
\item results in...
\end{itemize}

If you are asked to solve these equations, \( x = x + 1 \) and \( x + (-x) = 0 \), your answers should be \( \emptyset \) and \( \mathbb{R} \).

\begin{itemize}
\item \texttt{\textbackslash cents \textbackslash degrees \textbackslash inches \textbackslash feet}
\end{itemize}

These commands produce some common symbols which are missing from your keyboard. The names are self-explanatory, and the example below illustrates their use.

Note: The symbols are given plural names so that they are easier to remember. It may feel awkward to type, say, \texttt{1\textbackslash degrees}, but that will look just fine in print. Also, notice there is no space between the degree symbol and the ‘F’ in the typeset output. This is because TeX ignores the space after the control-word. If you wanted to insert some extra space, you could type \texttt{\textbackslash degrees\textbackslash F}. The double-backslash would insert a thinspace and prevent a bad line break from occurring. The technique is used in the third example to separate feet and inches (you could use a hyphen instead).
Example

Find the unit price for beef jerky, if 3 oz costs 75 cents.

Convert 138 degrees F to Celsius.

What is the area of rectangle which measures 3 feet 6 inches by 4 feet 4 inches?

results in...

Find the unit price for beef jerky, if 3 oz costs 75¢.

Convert 138° F to Celsius.

What is the area of rectangle which measures 3' 6" by 4' 4"?

\texttt{\textbackslash rpos \textbackslash rneg}

These commands produce raised positive and negative signs, which may be appropriate for elementary mathematics classes (when introducing signed numbers). The commands work the same in text and math modes. Normally, they should be followed by a number or expression. If they are used with variables, or mixed up with regular plus and minus signs, then math mode should be turned on at least temporarily (this will assure proper spacing).

Example

What is the sum of \texttt{\textbackslash rneg}7 and \texttt{\textbackslash rpos}7?

Simplify: \$ 10x+\texttt{\textbackslash rneg}5x+\texttt{\textbackslash rneg}3x \$.

results in...

What is the sum of \texttt{\textbackslash rneg}7 and \texttt{\textbackslash rpos}7?

Simplify: 10x + \texttt{\textbackslash rneg}5x + \texttt{\textbackslash rneg}3x.

\texttt{\textbackslash seg \textbackslash ray \textbackslash lin \textbackslash arc}

These commands produce segment, ray, line and arc symbols. Each command takes a single parameter, as in \texttt{\textbackslash seg\{endpoints\}}. Most often, the endpoints will consist of two uppercase letters, but you can use three letters to signify a major arc (e.g., \texttt{\textbackslash arc\{ABC\}}). In all cases, you need to put grouping symbols around the points.

It may be tempting to use the command \texttt{\textbackslash line} instead of \texttt{\textbackslash lin}, but the former has a very different meaning to \TeX. (\TeX's line is made up of letters and things, not an infinite number of points.) You will have an easier time remembering \texttt{\textbackslash lin} if you observe that it has the same number of characters as \texttt{\textbackslash seg, \textbackslash ray and \textbackslash arc}.
Example

Given $\overrightarrow{AB}$ and $\overrightarrow{AC}$, must $\overrightarrow{AB}$ contain $\overrightarrow{AC}$?

If D, E, and F are three points on a circle, can you conclude that $m\arc{DE} + m\arc{EF} = m\arc{DEF}$?

results in...

Given $\overline{AB}$ and $\overline{AC}$, must $\overline{AB}$ contain $\overline{AC}$?

If D, E, and F are three points on a circle, can you conclude that $m\arc{DE} + m\arc{EF} = m\arc{DEF}$?

\[\angle \triangle \circle \quadr \llgram\]

These commands produce geometric symbols. The first three have obvious meanings, while the other two stand for quadrilateral and parallelogram. The commands are slightly unusual, because they have an optional parameter. You may follow any of them with a group of points, just as you do for segments, rays, etc. Or you can skip the points—e.g., you can produce the $\angle$ symbol by typing \llgram.

In case you were wondering, the reason \quad is not assigned to quadrilaterals is that the author of \TeX got to the word first. It is a typesetter’s term for one “em” of space, and that traditional meaning is preserved in the ACC format (for more information, refer to the previous section).

Note: In the example, ties are used to prevent bad line breaks. A tie is especially important when a single letter (i.e., a point) is followed by a punctuation mark. To understand why, imagine a letter and a period at the start of a new line; it would look like a subpart or multiple-choice answer rather than a continuation of the previous line.

Example

In $\triangle{ABC}$, $m\angle A + m\angle B = m\angle C$.

What kind of triangle is it?

$\circle P$ refers to a circle whose center is $P$.

$\quadr EFGH$ refers to a quadrilateral whose vertices are E, F, G and $H$.

results in...
In △ABC, m∠A + m∠B = m∠C. What kind of triangle is it?

□P refers to a circle whose center is P.

□EFGH refers to a quadrilateral whose vertices are E, F, G and H.

\textbf{\textbackslash squared  \textbackslash cubed}

These commands are convenient ways to produce exponents in text mode. Although they operate the same in math mode, they have no purpose there; it is easier just to type \(^2\) or \(^3\). The idea behind the commands is that they allow you to type dimensions as ordinary text, such as 5\(\text{km}\textbackslash squared \) (which appears in print as 5km\(^2\)). If you do not like this approach, you have the option of turning on math mode and typing the exponents normally.

\textbf{Example}

A “cc” is an abbreviation for one cubic centimeter (cm\textbackslash cubed).\par

A “hectare” is the same as 10,000\(\text{m}\textbackslash squared\).

results in...

A “cc” is an abbreviation for one cubic centimeter (cm\(^3\)).

A “hectare” is the same as 10,000\(\text{m}\(^2\).

\textbf{\textbackslash tf (text fraction)}

The command \textbackslash tf has a somewhat unusual operation. What it does depends on your preference for “text fractions.” When you run Acces, you decide whether the fractions should be small, large or diagonal. Here is an example of each:

small → 4½ hours \quad large → 66²₃\% \quad diagonal → ⅞ km

The command has two parameters, as in \(\textbackslash tf\{\text{numerator}\}\{\text{denominator}\}\). If the numerator or denominator is more than a single character, you need to put grouping symbols around it.

The name of the command is a little misleading, because the fraction may appear in text or math modes. Yet its purpose is quite narrow. It is meant to be used in text-like situations, where there will be no confusion about what is in the numerator or denominator, especially when a diagonal fraction is selected. If you want to use \textbackslash tf in math, you should limit it to numerical coefficients; in that case, the command will behave much like \textbackslash f. (For more information about fractions in math mode, see next section.)
Note: The examples demonstrate small, large and diagonal text fractions in that order. In the second example, the text fraction is typed in math mode because of the negative sign. Also, braces are not needed around the ‘6’ but are there for consistency.

Example

What is $33\frac{1}{3}\%$ of 126?
Find the average of $\frac{21}{10}$ and $-\frac{6}{10}$.
A man hikes $\frac{3}{4}$ mile in $\frac{1}{2}$ hour. What is his rate?

results in...

What is $33\frac{1}{3}\%$ of 126?
Find the average of $\frac{21}{10}$ and $-\frac{6}{10}$.
A man hikes $\frac{3}{4}$ mile in $\frac{1}{2}$ hour. What is his rate?

\[\alpha...\Omega\]

These commands produce Greek letters (see Appendix A for a complete list). The idea is simple: each command is a backslash followed by the name of the Greek letter. Thus, if you type $(\alpha,\beta,\gamma)$, you will see in print $(\alpha,\beta,\gamma)$. The only thing that distinguishes upper- from lowercase letters is the first character in their names. For example, $\phi$ and $\Phi$ produce $\phi$ and $\Phi$.

Note: The second example uses the command $\sin$ to produce a sine function. Why is the command needed? Briefly, it typesets “sin” with normal, upright letters; without it, the letters would be italicized and they would look like variables (more details are given in the next section).

Example

Here are some simple formulas that use Greek letters:
$\omega=\theta/t$, $E=\sigma/\epsilon$, and $c=\lambda\nu$.

The Biot–Savart Law,
$\delta B= \mu_0 l \delta L \sin(\alpha/4\pi r^2)$,
is used to calculate the magnetic flux around an electric wire.

results in...

Here are some simple formulas that use Greek letters: $\omega = \theta/t$, $E = \sigma/\epsilon$, and $c = \lambda\nu$.

The Biot–Savart Law, $\delta B = \mu_0 \delta L \sin(\alpha/4\pi r^2)$, is used to calculate the magnetic flux around an electric wire.
A few Greek letters have variant forms. To obtain these letters, you simply precede their names with “var.” Thus, the commands (\phi, \theta, \epsilon, \rho) and (\varphi, \vartheta, \varepsilon, \varrho) yield (\phi, \theta, \epsilon, \rho) and (\varphi, \vartheta, \varepsilon, \varrho).

\textbf{Example}

Some people like to represent electric permittivity with the symbol $\epsilon$. Other people prefer $\varepsilon$.

\textit{results in...}

Some people like to represent electric permittivity with the symbol $\epsilon$. Other people prefer $\varepsilon$.

\textbf{Example}

A puzzle: The area of a rectangle is given by the formula $A=\ell w$. Is there anything wrong with saying $A=6$ when $\ell=2$ and $w=3$?

\textit{results in...}

A puzzle: The area of a rectangle is given by the formula $A=\ell w$. Is there anything wrong with saying $A=6$ when $\ell=2$ and $w=3$?
TEX \texttt{\textbackslash infty}

This command produces the infinity symbol $\infty$, which is commonly used with limits, derivatives, and other advanced formulas. In Plain TeX, the command is restricted to math mode. In the ACC format, it can be used in ordinary sentences as well. This is useful if you want to type something like

Observe the height as $t$ approaches \texttt{\textbackslash infty}

which appears in print as

Observe the height as $t$ approaches $\infty$.

Note: The example below uses the command \texttt{\textbackslash lim} to produce a limit function. The actual limit, $x \to \infty$, is easy to obtain; it is formed by the subscript character $_$. (For more information about upper and lower limits, refer to the command \texttt{\textbackslash int} in the next section.)

\textbf{Example}

$ \lim_{x \to \infty} \frac{1}{x^2} + 6 = 3 $  

\textit{results in}...

\[ \lim_{x \to \infty} \frac{1}{x^2} + 6 = 3 \]

TEXT \texttt{\textbackslash squareans}

This command produces a large empty square, which is sometimes used in elementary math texts as a way of introducing variables. (It may serve other purposes, as well.) The command can be used in math or text modes.

\textbf{Example}

What number is missing from this equation: $3 \texttt{\textbackslash squareans}^2 = 48$?

Fill in the squares: $\texttt{\textbackslash squareans}\texttt{\textbackslash squareans} \times 13 = 156$.

\textit{results in}...

What number is missing from this equation: $3 \square \square^2 = 48$?

Fill in the squares: $\square \square \times 13 = 156$. 

\begin{center}
\textit{Summary of Commands}
\end{center}

45
Math formulas

\TeX - (minus sign)

In math mode, the - character produces a minus sign (or negative sign, depending on how you use it). Do not mix up this sign with a hyphen or dash. There is more space around a minus. Also, it is longer and wider than its relatives in text mode.

Note: The example uses the characters - + and =. There is nothing unusual about the latter two, so they are not listed as commands. Also, if you look at various minus signs in the example, you will see how \TeX makes subtle spacing adjustments.

\begin{itemize}
  \item Example
  \end{itemize}

Solve: $ -3y-(-y+6)=4y-4 $ 

results in...

Solve: $-3y - (-y + 6) = 4y - 4$

\TeX _ and ^

These commands, which are active characters, produce sub- and superscripts. Each one takes a single parameter, as in _⟨math material⟩_. The math material is automatically lowered or raised and typeset in a smaller font. Be sure to use grouping symbols if the material consists of more than one character.

\begin{itemize}
  \item Example
  \end{itemize}

$ (x^3y^{-2})^4 $ \quad $ P=P_0(1+r)^t $ \quad $ T=T_ie^{-0.01x} $ 

results in...

$ (x^3y^{-2})^4 \quad P = P_0(1 + r)^t \quad T = T_ie^{-0.01x} $
In math mode, these commands produce symbols for multiplication and division (\(\cdot\), \(\times\), and \(\div\)). Even though the commands are made up of single letters, they are still control words. Be careful when they are followed by letters (variables). You will need to type a space in between.

Note: In the second example, the only space that is truly needed is the one after \(\times\). Without it, \TeX{} would think you were giving the command \(\times y\). (That command does not exist and would be ignored.) Also, do you see why there are no parentheses or spaces after \(\div\)? There are two reasons: both the numerator and denominator are single characters, and \TeX{} knows the ‘1’ is not part of the command.

\textbf{Example}

What is \(3.45 \times 10^{-4}\) in standard decimal notation?

Is \(y \times y\) the same as \(y \div \frac{1}{y}\)?

results in...

What is \(3.45 \times 10^{-4}\) in standard decimal notation?

Is \(y \times y\) the same as \(y \div \frac{1}{y}\)?

In math mode, the character combinations \(\geq\), \(\leq\), \(\neq\), \(\gg\) and \(\ll\) produce the inequalities \(\geq\), \(\leq\), \(\neq\), \(\gg\) and \(\ll\). The first three combinations are used in computer programming, so you may be familiar with them already. The latter two mean “much less” and “much greater.”

\textbf{Example}

Graph each solution:

\(\indent 4x \geq -8 \quad \text{and} \quad -7x \neq 3x - 10\)

\(\indent 3y - 5 < 3 - y \leq 6y\)

results in...

Graph each solution:

\(4x \geq -8 \text{ and } -7x \neq 3x - 10\)

\(3y - 5 < 3 - y \leq 6y\)
These commands produce the symbols $\approx$, $\doteq$, $\propto$, $\cong$ and $\sim$. You may be tempted to use them in ordinary text, especially in geometry problems. You must resist the temptation. All of the symbols are mathematical relations and must be treated as such.

**Example**

\[
\begin{align*}
\triangle ABC & \sim \triangle DEF \\
\triangle MNO & \approx 30^\circ
\end{align*}
\]

results in...

\[
\begin{align*}
AB & \cong CD \\
m\angle MNO & \approx 30^\circ \\
\triangle ABC & \sim \triangle DEF
\end{align*}
\]

This command produces the symbol $\eq$, which is used in some secondary math books. The symbol is considered to be a relation, though it really poses a question: “Is the equality true?”

**Example**

\[
\begin{align*}
\frac{12}{45} & \eq \frac{15}{4}
\end{align*}
\]

results in...

\[
\begin{align*}
12 \div 15 \\
\frac{12}{45} & = \frac{15}{4}
\end{align*}
\]

These commands produce the symbols $\cup$ and $\cap$. They are synonyms for the Plain \TeX commands \texttt{cup} and \texttt{cap}. The latter pair may be easier to type, the former easier to remember.

**Example**

\[
A \cup (B \cap C)
\]

results in...

\[
A \cup (B \cap C)
\]
These commands produce $\parallel$ and $\perp$ signs. The first one is an abbreviated form of the plain \TeX command $\parallel$, which you may like to use instead.

**Example**

\[
\ell_1 \parallel \ell_2 \quad \text{and} \quad \overrightarrow{GH} \perp \overrightarrow{JK}
\]

This command negates a relationship. The formal usage is $\not(math\ relation)$. Basically, all the command does is typeset a forward slash on top of symbol.

Note: The $\not$ command works fine with all symbols, although two special cases exist. To get $\ne$, you can type $\langle$, $\ne$ or $\not=$. To get $\notin$, it is recommended you type the single command $\not\in$ instead of $\not\in$ (the former produces a specially designed symbol, which looks better).

**Example**

\[
\overrightarrow{AB} \not\perp \overrightarrow{BC} \quad A \not\subset B
\]

These commands do what their names imply: they put a line over or under a formula. Each takes a single parameter, as in $\overline(math\ material)$. If the math material is more than a single character, you need to put it inside a group.

**Example**

\[
1.\overline{3} \quad 0.0\overline{52} \quad \overline{3+4i}
\]
$ \textbackslash{ldiv}$

This command produces a long division sign. The usage is $\textbackslash{ldiv}\langle\text{math material}\rangle$. Properly speaking, the math material is the “dividend.” If it is more than a single character, you need to put it inside a group. You will probably want to type a “divisor” in front of the command, but you are not required to do so.

$ 0.24 \textbackslash{ldiv} 5 \textbackslash{quad} x-1\textbackslash{ldiv}\{2x^3-x+1\} $ results in...

\begin{align*}
0.24 \div 5 \quad x-1 \div 2x^3 - x + 1
\end{align*}

$\textbackslash{f} \textbackslash{df} \textbackslash{pf} \textbackslash{xf}$

These commands produce fractions in math mode. Each of them takes two parameters, as in $\textbackslash{f}\langle\text{numerator}\rangle\langle\text{denominator}\rangle$. If the numerator or denominator is more than a single character, you need to put it inside a group. Here is an illustration of each command, based on the expression “$a$ over $b$”:

$\textbackslash{f} \rightarrow \frac{a}{b} \quad \textbackslash{df} \rightarrow \frac{a}{b} \quad \textbackslash{pf} \rightarrow \left(\frac{a}{b}\right) \quad \textbackslash{xf} \rightarrow \frac{a}{b}$

The two most important commands are $\textbackslash{f}$ and $\textbackslash{df}$. The former produces an ordinary fraction, which varies in size. The latter produces a “display style” fraction, which is always large. The difference between the commands is subtle—the best choice depends on the context and how much of a perfectionist you are.

We should point out that all fractions increase in size if the numerator or denominator increases. What makes $\textbackslash{f}$ special is that $\TeX$ also considers the location of the fraction. If you type $\textbackslash{f}\{\text{f12}\}$ you will see a medium and small fraction in print: $\frac{1}{2^{\frac{1}{2}}}$. Of course, this variation in size is exactly what you want. It makes the command $\textbackslash{f}$ a good choice for exponents and coefficients, mixed numbers, and simple proportions.

In some situations, you may want a larger fraction than $\textbackslash{f}$ provides. That is the purpose of the command $\textbackslash{df}$. We recommend it for advanced algebraic equations, trigonometric identities, and complicated formulas. The only place where $\textbackslash{df}$ is not appropriate is in superscripts and subscripts. As for the remaining commands, $\textbackslash{pf}$ puts parentheses around a display style fraction (it is equivalent to typing $\textbackslash{p}\{\text{df} \ldots\}$), and $\textbackslash{xf}$ is used for extended or complex fractions (it produces a slightly thicker fraction bar than $\textbackslash{df}$).
Note: Each example has some numerators and denominators inside curly braces.
If you are not concerned about short-cuts, or if you just want to be consistent, you can use grouping symbols all of the time (they don’t do any harm with fractions).

**Example**

Graph: \( y = -\frac{5}{4}x + 6 \) \rt and \rt \( \frac{10}{3}(y-1) = (x+2) \)

Simplify each of these expressions:

\[
\frac{1}{1 + \frac{1}{x}} \quad \frac{\frac{1}{2} + \frac{1}{3}}{\frac{1}{3} + \frac{1}{6}} \quad \left( \frac{x^2(-2x^3y^5)}{-y^4(xy)} \right)^{\frac{1}{2}}
\]

results in . . .

Graph: \( y = -\frac{5}{4}x + 6 \) and \( \frac{10}{3}(y - 1) = (x + 2) \)

Simplify each of these expressions:

\[
\frac{1}{1 + \frac{1}{x}} \quad \frac{\frac{1}{2} + \frac{1}{3}}{\frac{1}{3} + \frac{1}{6}} \quad \left( \frac{x^2(-2x^3y^5)}{-y^4(xy)} \right)^{\frac{1}{2}}
\]

These commands produce radical expressions: square roots and indexed roots. The usage for the first is \( \sqrt{\text{math material}} \). The usage for the second is \( \sqrt{\text{index}}(\text{math material}) \). In both cases, the radical will grow as large as necessary to encompass the math material or, more precisely, the “radicand.” As always, you need to use grouping symbols if the index or radicand is more than a single character.

Note: As the second example shows, you can put a radical inside a fraction or a fraction inside a radical. Just be sure the braces match up correctly.

**Example**

Find the sum of \( 3\sqrt{2} \) and \( -\sqrt{50} \).

\[
\sqrt{\frac{2}{3}} \quad \sqrt{\frac{24}{18}} \quad \sqrt[3]{128x^9y^7}
\]

results in . . .

Find the sum of \( 3\sqrt{2} \) and \( -\sqrt{50} \).

\[
\sqrt{\frac{2}{3}} \quad \sqrt{\frac{24}{18}} \quad \sqrt[3]{128x^9y^7}
\]
These commands produce delimiters around a math formula: parentheses, brackets, braces and vertical bars. The size of the delimiters will vary, depending on the height and depth of the formula. The last command, with the uppercase V, produces double vertical bars, as in $\| x \|$. Its usage is $\texttt{\textbackslash Verts} \langle \text{math material} \rangle$. The other commands, which produce $()$, $[ ]$, $\{ \}$ and $\mid$, are used in a similar manner.

\TeX{} tends to be conservative about increasing the size of delimiters. $\texttt{\textbackslash parens}$ and $\texttt{\textbackslash bracks}$, for example, do not have a noticeable effect unless they act on a tall formula. If a formula does not contain an exponent, fraction or radical, you might just as well use ordinary parentheses or brackets (those found on the keyboard). The same advice does not apply to curly braces and vertical bars for the following reasons.

- Curly braces cannot be obtained by typing \{ and \}, because those characters are used as grouping symbols. If you want to print the characters themselves, you have to give the \TeX{} commands $\texttt{\textbackslash lbrace}$ and $\texttt{\textbackslash rbrace}$ or \{ and \}. Thus we recommend the one command $\texttt{\textbackslash braces}$: it takes less effort to type and it produces delimiters which vary in size.

- Vertical bars (as delimiters) cannot be obtained by typing ‘|’, because the keyboard character has no left or right orientation. The character works fine if you want to insert a bar in the middle of a formula. But when it comes to putting bars around a formula, the commands $\texttt{\textbackslash verts}$ and $\texttt{\textbackslash Verts}$ produce better-looking results: the bars vary in size and they are recognized by \TeX{} as real opening and closing marks.

While \TeX{} can adjust the size of delimiters automatically, its methods are not perfect. Once in a while, something that is typeset according to rules just doesn’t look right. There are ways of overriding \TeX{} or adjusting each delimiter by hand, but those topics are a little too advanced for this discussion. Please see a \TeX{} reference book for more information.

Note: The commands $\texttt{\textbackslash parens}$, $\texttt{\textbackslash braces}$, etc., may be nested to any number of levels, as the first example demonstrates. Also, they may encompass fractions or radicals, or vice-versa.
Example
\[ \sqrt{-7-(2-5)}\{2+(7-5)\} \quad \text{\textasciitilde} \quad \sqrt{2+(7-5)} \]
\[ \left( (k^2-1)^2 - 1 \right)^2 \]
Evaluate \[ y^2-(y-1) \] for \( y = \{-1, 0, \frac{1}{2}, \frac{2}{3}, 2\} \).

results in...
\[ \left\{ \frac{-7-(2-5)}{2+(7-5)} \right\} \quad \left( \left( (k^2-1)^2 - 1 \right)^2 - 1 \right) \]

Evaluate \[ y^2-(y-1) \] for \( y = \{-1, 0, \frac{1}{2}, \frac{2}{3}, 2\} \).

\[ \left\{ \frac{-7-(2-5)}{2+(7-5)} \right\} \quad \left( \left( (k^2-1)^2 - 1 \right)^2 - 1 \right) \]

Example
\[ \sqrt{x^2} = \abs{x} \]

Solve \[ \sqrt{(x+1-1)(\sqrt{x+1}+1)} = x \] for \( x \geq 0 \).

results in...
\[ \sqrt{x^2} = \abs{x} \]

Solve \( (\sqrt{x+1}-1)(\sqrt{x+1}+1) = x \) for \( x \geq 0 \).

\[ \antilog... \log... \tan \]

These commands produce functions in math mode (a complete list is given in Appendix A). They do not take any parameters, although they can be used with limits—i.e., superscripts and subscripts. The commands serve two purposes. First, they typeset the names of the functions in normal, upright letters (instead of math italics). Second, they put a little extra space between each function and its argument, which is proper in mathematics.

Since the function commands are made up of letters, you must be careful when they are followed by another letter (variable). A common technique is to put a space after the command, as in \texttt{\sin x}. You may also put the variable inside a group, as in \texttt{\sin\{x\}}. If the command is followed by anything other than a letter, no spaces or grouping symbols are needed. For example, it is perfectly okay to type \texttt{\sin\alpha}.

Summary of Commands 53
Note: The first example illustrates a fine point about mathematical typesetting. If the question marks were not put in curly braces, \TeX{} would butt them up against the equal signs, thinking they were punctuation marks. The curly braces tell \TeX{} to treat them as ordinary math symbols, which produces the right amount of space after the equal signs.

\begin{itemize}
  \item Example
  \begin{align*}
    \log_2(64) &= {}? \quad e^{\ln 100} = {}? \\
    \csc x \cdot \tan x &= \sec a \\
    \arccos(0.866) &\approx \frac{\pi}{6} \\
    \cos^{-1}\theta &= \frac{1}{\cos \theta}
  \end{align*}
\end{itemize}

results in…

$$\log_2(64) = {}? \quad e^{\ln 100} = {}? \quad \csc x \cdot \tan x = \sec a \quad \arccos(0.866) \approx \frac{\pi}{6} \quad \cos^{-1}\theta = \frac{1}{\cos \theta}$$

These commands are used to write permutations and combinations. Each takes two parameters, as in \P{\text{group number}}{\text{subgroup number}}. The commands behave something like “text fractions,” in that their results are based on a particular style, which the end-user can select inside of \textit{Acces}.

Note: In the example, the permutation and combination are printed in the default style, which is \textit{nPr}. You can change this to \textit{nPr} or \textit{P(n,r)} (see document layout).

\begin{itemize}
  \item Example
  \begin{align*}
    \P{n}{r} &= \frac{n!}{(n-r)!} \quad \C{10}{2} = 45
  \end{align*}
\end{itemize}

results in…

$$\P{n}{r} = \frac{n!}{(n-r)!} \quad \C{10}{2} = 45$$
\textbf{\texttt{\textbackslash pick}}

This command provides an alternate way of writing a combination. Just like \texttt{\textbackslash c}, it takes two parameters; however, a combination formed by \texttt{\textbackslash pick} always appears as a binomial coefficient, regardless of the current style. In other words, if you type \texttt{\textbackslash pick 5 2} you will see $\binom{5}{2}$ in print.

**Example**

$$ (a+b)^n = \texttt{\pick n 0 a \textasciicircum n b \textasciicircum 0} + \texttt{\pick n 1 a \textasciicircum{n-1} b \textasciicircum 1} + \texttt{\pick n 2 a \textasciicircum{n-2} b \textasciicircum 2} + \cdots + \texttt{\pick n{n-1} a \textasciicircum 1 b \textasciicircum{n-1}} + \texttt{\pick n n a \textasciicircum 0 b \textasciicircum n} $$

results in...

$$(a + b)^n = \binom{n}{0}a^nb^0 + \binom{n}{1}a^{n-1}b^1 + \binom{n}{2}a^{n-2}b^2 + \cdots + \binom{n}{n-1}a^1b^{n-1} + \binom{n}{n}a^0b^n$$

\textbf{\texttt{\textbackslash vec \textbackslash Vec}}

These commands are used to denote vectors. The first puts a vector accent on top of a letter, like $\vec{a}$. The second typesets the letter in a bold face, like $\mathbf{a}$. Each command takes one parameter, as in \texttt{\vec{math material}}. Normally, the math material will be a single lowercase letter, so you do not need to use grouping symbols. You should be careful, however, to put a space between the command and the letter.

**Example**

$$ \texttt{\abs{\vec{v}}} = \sqrt{v_x^2 + v_y^2} $$

$$ \texttt{\Vec a} = -2\mathbf{i} + 3\mathbf{j} $$

results in...

$$ |\vec{v}| = \sqrt{v_x^2 + v_y^2} $$

$$ \mathbf{a} = -2\mathbf{i} + 3\mathbf{j} $$
\texttt{\textbackslash to \textbackslash rightarrow}

These commands produce the symbol $\to$. They are exact synonyms: the first is easier to type; the second is perhaps easier to remember. Many other arrow commands are available (see Appendix A for a complete list).

Since TeX treats arrows as relations, commands like \texttt{\textbackslash to} and \texttt{\textbackslash rightarrow} are restricted to math mode. If you want the arrow to appear in text, you can turn on math mode temporarily.

\textbf{Example}

\begin{verbatim}
\[
\lim_{x \to 0} \frac{\sin x}{x} = 1
\]
\end{verbatim}

results in...

\begin{equation}
\lim_{x \to 0} \frac{\sin x}{x} = 1
\end{equation}

\texttt{\textbackslash int \textbackslash Int \textbackslash sum \textbackslash Sum}

In math mode, these commands produce integral and summation signs. As you might guess, the first letter determines the size of each operator. Here are examples of the commands:

\begin{align*}
\texttt{\textbackslash int} & \quad \texttt{\textbackslash Int} \quad \texttt{\textbackslash sum} \quad \texttt{\textbackslash Sum}
\end{align*}

Limits are easily added to integrals and summations. To typeset an upper limit, use the control character ^\texttt{\textbackslash \_\_}. To typeset a lower limit, use the character _\texttt{\textbackslash \_\_}. The rules are the same as with superscripts and subscripts; in other words, if a limit is more than a single digit or letter, put it inside curly braces. The order in which limits are given does not matter.

Note: You should use small operators, \texttt{\textbackslash int} and \texttt{\textbackslash sum}, in text situations (remember to turn on math mode temporarily). You may also want to use small operators where there are no limits. Large operators tend to look better when limits are present.
Example

By default, \TeX\ places limits to the right of a $\int$ and above and below a $\sum$.

\[
\int_0^\infty \frac{1}{x} \, dx \quad \sum_{i=1}^{10} x^2
\]

results in...

By default, \TeX\ places limits to the right of a $\int$ and above and below a $\sum$.

\[
\int_0^\infty \frac{1}{x} \, dx \quad \sum_{i=1}^{10} x^2
\]

\TeX\ \texttt{\ldots, \cdots, \vdots, \ddots}

These commands, which produce three dots, imply a continued pattern. The position and direction of the dots is based on the first letter: \texttt{l} = lower, \texttt{c} = center, \texttt{v} = vertical and \texttt{d} = diagonal.

Note: ‘\ldots’ and ‘\cdots’ are commonly used in sequences and series. The others are used mainly to describe a matrix (see the command \texttt{\matrix} for an example).

Example

\[
1, 1, 2, 3, 5, 8, \ldots, (x_{n-1}+x_{n-2})
\]

\[
\sum_{1}^{10} x^2 = 1 + 4 + 9 + \cdots + 100
\]

results in...

1, 1, 2, 3, 5, 8, \ldots, (x_{n-1} + x_{n-2})

\[
\sum_{1}^{10} x^2 = 1 + 4 + 9 + \cdots + 100
\]
This command is used to write a system of equations. While this can be done merely by beginning a new line (see below), the command \texttt{\simuleq} has the advantage of lining up the left and right sides of the equations, which is often desirable.

The proper usage is \texttt{\simuleq\{alignment entries\}}. Alignment entries take different forms, depending on how a command is defined. With \texttt{\simuleq}, the entries must be put between a pair of curly braces and they must follow the pattern

\begin{align*}
\text{left side of equation} & \text{=} \text{right side of equation} \cr
\end{align*}

The \texttt{\cr} stands for carriage return. It is required after every row except the last. The character \texttt{&} tells TeX where to align things. While TeX is capable of lining up individual terms, \texttt{\simuleq} does not provide a means of doing this.

\texttt{\simuleq} is intended for math mode, but not restricted to it. If the command is used in text mode, the alignment entries are still treated as math formulas.

If you do not want to line up equations by their equal signs, a much simpler method is available. All you need to do is start a new line after each one. In other words, you can type something like

\begin{align*}
equation 1 \\text{\textbackslash dn\newline} \equation 2 \\text{\textbackslash dn\newline} \equation 3 \ldots
\end{align*}

This method works quite well, and sometimes it looks better than the results of \texttt{\simuleq}.

Note: Since alignment entries are treated as math formulas, spaces are irrelevant. Also, it makes no difference to TeX whether the entries span one or several lines. As the second example shows, it is possible to spread things out and do a rough approximation of the alignment. By the way, the command \texttt{\simuleq} works just as well with inequalities.
Example

(1) \( \simuleq{ 2x + 3y &= -6 \cr 4x - \frac{1}{2}y &= 10 \cr } \quad \text{\quad quad quad} \)

(2) \( \simuleq{ x + y &= 0 \cr y + z &= 1 \cr x + y + z &= 2 \cr } \)

results in...

(1) \( 2x + 3y = -6 \)
\( 4x - \frac{1}{2}y = 10 \)

(2) \( x + y = 0 \)
\( y + z = 1 \)
\( x + y + z = 2 \)

\( \text{\%} \cases \)

This command is used for multiple-part functions or case statements. The proper usage is \( \cases{ (alignment\ entries) \} \). The entries must be put between a pair of curly braces and they must follow the pattern:

\( \text{formula \& condition}\cr \)

The \( \text{\quad cr} \) command is required after every row except the last. The alignment character \& is used to separate formulas and conditions.

Note: The formulas that appear in case statements are assumed to be mathematical (they are implicitly surrounded by dollar signs). Conditions, on the other hand, are treated as ordinary text, because they contain words like “for”, “else” and “otherwise.” When specifying a condition, you can still turn math mode on temporarily, as the example shows.

Example

\( \$ g(x) = \cases{ x^2 & for \ x \leq 0, \cr x+3 & for \ 0 < x < 4, \cr -x & otherwise. \cr } \) \$

results in...

\( g(x) = \begin{cases} 
  x^2 & \text{for } x \leq 0, \\
  x + 3 & \text{for } 0 < x < 4, \\
  -x & \text{otherwise.} 
\end{cases} \)
This command lets you build a matrix of any size. The proper usage is \( \text{\textbackslash matrix\{\langle alignment\ entries\rangle\} \), where each row of the alignment takes the form:

\[
\begin{align*}
entry 1 & \ & entry 2 & \ & entry 3 & \ldots \text{\textbackslash cr}
\end{align*}
\]

The \textbackslash cr is required after each row except the last. The alignment character \& separates one entry from the next.

\TeX\ does not put any delimiters around a matrix, so that the command remains general-purpose. It can be used to illustrate the addition of polynomials, to line up the terms of equations, and for a surprising number of non-math applications. (Some people use the command \texttt{\textbackslash matrix} to diagram chemical reactions.) If you want delimiters, you can type something like \texttt{\textbackslash bracks\{\textbackslash matrix\{\ldots\}\}} or you can use a short-cut command like \texttt{\textbackslash bmatrix} (see below).

Note: Each entry in a matrix is treated as a math formula (it is implicitly surrounded by dollar signs). Since spaces are ignored in math mode, you can spread things out and represent the matrix in raw form. Also, the example shows a square matrix, but the number of columns and rows does not have to be equal.

\begin{itemize}
\item \textbf{Example}
\end{itemize}

\[ \texttt{$ \textbackslash matrix\{ t\_{\{11\}} & t\_{\{12\}} & \cdots & t\_{\{1n\}} \text{\textbackslash cr} t\_{\{21\}} & t\_{\{22\}} & \cdots & t\_{\{2n\}} \text{\textbackslash cr} \vdots & \vdots & \ddots & \vdots \text{\textbackslash cr} t\_{\{m1\}} & t\_{\{m2\}} & \cdots & t\_{\{mn\}} \text{\textbackslash cr }$}$\]

results in...

\[
\begin{array}{cccc}
t_{11} & t_{12} & \cdots & t_{1n} \\
t_{21} & t_{22} & \cdots & t_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
t_{m1} & t_{m2} & \cdots & t_{mn} \\
\end{array}
\]
\determ \vmatrix \Vmatrix \bmatrix \pmatrix

These commands provide a short-cut method for putting delimiters around a matrix. They are used in exactly the same manner as \matrix; in other words, they each take one parameter, a group of alignment entries.

\determ and \vmatrix, which are synonyms, put vertical bars around a matrix. (The first command is an abbreviation for determinant.) \Vmatrix applies double vertical bars; it is the same as typing \Verts\matrix{...}. \bmatrix and \pmatrix apply brackets and parentheses, as you might have guessed.

\begin{itemize}
  \item Example
\end{itemize}

\begin{verbatim}
$ \determ{ \f12 & -3\f12 \cr 8 & 4 }$
\quad
$ \pmatrix{ \tan(\gamma) & \cos^2(\alpha) \cr \sin(\beta) & \sin^2(\beta) \cr \cos(\alpha) & \tan^2(\gamma) \cr }$
\end{verbatim}

results in...

\[ \begin{vmatrix}
\quad \begin{pmatrix}
\tan(\gamma) & \cos^2(\alpha) \\
\sin(\beta) & \sin^2(\beta) \\
\cos(\alpha) & \tan^2(\gamma)
\end{pmatrix}
\end{vmatrix} \]

\columnadd \columnsup \columnx

These commands are designed for primary- and secondary-level mathematics. They make it easy to write addition, subtraction and multiplication problems in columns. (Sometimes these are called vertical operations.) Each command takes a single parameter, as in \columnadd\{⟨expression1 expression2⟩\}. The expressions must be put inside a group and separated by spaces. Fractions and decimals are allowed; if present, they will be lined up automatically.

The commands are best-suited for simple arithmetic. If you use them with algebraic expressions, keep in mind the following: (1) spaces should not show up in any formulas (they are used as separators only); and (2) individual terms are not lined up automatically (that can be done, but the methods are beyond the scope of this manual).

While the commands are intended for math mode, they are not restricted to it. If they are used in text mode, the individual expressions will still be treated as math formulas (they are implicitly surrounded by dollar signs).
Example
\columnadd{200 92.3 7.84} \qquad \columnsub{10\f{12} 2\f{34}} \qquad \columnx{x^2+x+5 x-5}

results in...
\[
\begin{array}{ccc}
200 & 10\frac{1}{2} & x^2 + x + 5 \\
92.3 & -2\frac{3}{4} & \times x - 5 \\
+ 7.84 & & \\
\end{array}
\]
Special effects

\smallblank \medblank \bigblank

These commands insert blank (underlined) spaces in text or math mode. They are used for “fill-in-the-blank” questions, among other things. The commands are not affected by font size. Here is the result of each:

\smallblank (0.3 inches) → ____
\medblank (0.6 inches) → ________
\bigblank (1.2 inches) → _______________

No space is put on either side of the blanks. In some cases, it looks better to separate them from words, symbols or punctuation marks. You can use a control-space, tie or thinspace for that purpose (recall that the latter two commands prevent a line break from occurring).

Note: The examples below illustrate different spacing techniques. In the second paragraph, the tie joins together the medium blank and the word “degrees.” In the third paragraph, the control-space inserts a normal inter-word space after the command \degrees, and it allows a line break to occur. The thinspace at the end is not critical, but it distinguishes the period from the big blank and is a nice touch.

▷ Example

25 is \smallblank\% of 40.

The sum of the angle measures in a triangle is \medblank\degrees.

Two angles whose measures add up to 180\degrees\ are called \bigblank\.

results in...

25 is _____\% of 40.

The sum of the angle measures in a triangle is _______ degrees.

Two angles whose measures add up to 180° are called ____________________.
\texttt{\textbackslash blankfill \textbackslash blankline}

These commands go further than \texttt{\textbackslash smallblank}, \texttt{\textbackslash medblank} and \texttt{\textbackslash bigblank}. The first command fills the current line of text with blank (underlined) space. The second does exactly the same thing, except that it begins a new line; thus, the command can be used repeatedly. Although \texttt{\textbackslash blankfill} and \texttt{\textbackslash blankline} were designed for the end of a paragraph—for written answers—they may be used within a paragraph and in math mode.

Note: \texttt{\textbackslash blankfill} does not serve much purpose by itself. Its effect is mostly cosmetic and depends entirely on where line breaks occur. Normally the command should be followed by at least one \texttt{\textbackslash blankline}, so you are guaranteed a minimum amount of space for answers.

\begin{itemize}
\item \textbf{Example}
\end{itemize}

What is the difference between a combination and a permutation? \texttt{\textbackslash blankline \textbackslash blankline}

What is the difference between an algebraic and geometric series? \texttt{\textbackslash blankfill \textbackslash blankline}

\textit{results in…}

What is the difference between a combination and a permutation?

What is the difference between an algebraic and geometric series?

\texttt{\textbackslash blankfill \textbackslash blankline}

\texttt{\textbackslash subpart}

This command is used to write sub-paragraphs, or to break down a problem into several parts. There are basically two effects of the command: the first is to indent the entire sub-paragraph, and the second is to identify it with a letter or number. Although the counting or labeling mechanism is automatic, you can change the style (see \texttt{\textbackslash lettercount} below).

There are two restrictions on the command \texttt{\textbackslash subpart}: it must appear at the beginning of a paragraph, and it can be used in text mode only. You do not need to end the previous paragraph before typing \texttt{\textbackslash subpart}, since it contains an implicit \texttt{\textbackslash par} command. Also, you do not need to reset the counter (labeling mechanism) at the end of a problem, since changes are localized.
Note: As the example demonstrates, sub-paragraphs can be very short. You can use \subpart to identify a single word or mathematical expression, as well as several sentences. Also, you can indent subparts (in raw text), to represent them more clearly.

\begin{itemize}
  \item \textbf{Example}
  \begin{itemize}
    \item A rocket is fired straight up. After $t$ seconds, the rocket is $h$ feet above the ground, as given by the equation $h=96t-16t^2$.
    \item What is its height after
    \begin{itemize}
      \item $0.5$ seconds?
      \item $4$ seconds?
    \end{itemize}
    \item After how many seconds will the rocket be $112$ feet above the ground? (Round to nearest tenth.)
    \item Find the maximum height of the rocket.
  \end{itemize}
\end{itemize}

results in...

A rocket is fired straight up. After $t$ seconds, the rocket is $h$ feet above the ground, as given by the equation $h = 96t - 16t^2$. What is its height after

a) 0.5 seconds?
b) 4 seconds?

c) After how many seconds will the rocket be 112 feet above the ground? (Round to nearest tenth.)
d) Find the maximum height of the rocket.

\begin{itemize}
  \item \Lettercount \Lettercount \numbercount
  \end{itemize}

These commands change the way sub-paragraphs are labeled (a complete list is given in the section called \textit{document layout}). The first command is the default; it is implied at the beginning of every problem. It causes sub-paragraphs to be identified by lowercase letters in the form a), b)... If you like, you can override the default by typing \Lettercount or \numbercount before the first sub-paragraph. With those two commands, labeling takes the form A), B)... or 1), 2)...
results in...

Graph each equation on the same coordinate system.

A) \( y = x^2 \)
B) \( y = -3x^2 - 1 \)
C) \( y = \frac{1}{2}(x + 4)^2 \)

Draw a picture which illustrates:

1) line \( \ell_1 \) in horizontal plane \( R \)
2) line \( \ell_2 \), which intersects line \( \ell_1 \) and is perpendicular to plane \( R \)

\[ \text{EA} \ \mc \]

This command is used in multiple-choice problems. It labels an answer and (optionally) begins a new line. There are two ways of using the command. You can type \( \mc \) followed by some text or math material, or you can treat the command as having a parameter and put the material inside a group—that is, \( \mc{...} \).

- If you do not use curly braces, then \( \mc \) behaves a little like \( \text{subpart} \). In other words, the command begins a new line and puts a label in front of it. We recommend this approach when answers are lengthy or when they need to be distinguished as much as possible.

- If you use curly braces with \( \mc \), then each answer is typeset in normal paragraph mode. That is to say, the answer is moved to the right a little bit, but the line is not necessarily broken. (A line break will occur if the answer does not fit.) This approach works well with short answers: numbers, ratios, points, etc.

Note: Line breaks in your raw data do not affect the typesetting of answers. Thus, when writing multiple-choice problems, you can put each answer on a separate line. That makes the answers more readable and allows you to mix them up later.
Example

The ratio 1 foot to 20 inches is \(1:20\) \(20:1\) \(20:12\) \(5:3\) \(3:5\).

Two angles whose measures add up to 90\(\text{degrees}\) are called
\mc{vertical} \mc{adjacent} \mc{supplementary} \mc{complementary}

\textit{results in} . . .

The ratio 1 foot to 20 inches is  a) 1:20  b) 20:1  c) 20:12  d) 5:3  e) 3:5.

Two angles whose measures add up to 90\(\text{°}\) are called
a)  \mc{vertical}  
b)  \mc{adjacent}  
c)  \mc{supplementary}  
d)  \mc{complementary}

\texttt{\emcee}

The command \texttt{\emcee} represents one of \textit{Acces'} most advanced features. It also shows the advantages of generalized markup, or the dynamic formatting of data. Since multiple-choice answers are not formatted until they are pulled out of the database, it is possible for them to be re-ordered and labeled in a variety of ways. That is the purpose of \texttt{\emcee}, the multiple-choice scrambler.

The proper usage for the command is \texttt{\emcee\{multiple choice answers\}}. There are three rules to keep in mind: 1) the answers must be surrounded by curly braces; 2) each answer must appear on its own line and be preceded by the command \texttt{\mc}; and 3) the correct answer should be marked by an asterisk. The asterisk is not actually required, but it does allow \texttt{\emcee} to track the right answer, in case scrambling is turned on.

\texttt{\emcee} can handle as many as five choices. We recommend that you use the command for all multiple-choice problems, unless you have a specific reason not to. It gives end-users much more flexibility. When running \textit{Acces}, they can decide how answers should be typeset and whether scrambling should be turned on. You can override these options at the problem-writing stage; however, as a general rule, you should allow such decisions to be made at “run-time.”
Note: In the example, the same problem is printed twice, as if it were re-selected inside the *Acces* program. The first time, scrambling is turned off and the *emcee* layout is optimized. The second time, scrambling is turned on with the option “none of the above” and the *emcee* layout is vertical. Also, notice that dollar signs surround the choices—this is an excellent short-cut if all five choices are math formulas. You will find many more hints for in the sample data files that come with *Acces*.

**Example**

What is the equation of the line that contains $(-1,0)$ and $(2,3)$?

\emcee{$$
*\text{mc } y=x
\text{mc } y=-x+1
\text{mc } y=3x+1
\text{mc } y=x+\frac{1}{3}
\text{mc } y=\frac{1}{3}x-1
$$}

results in...

1. What is the equation of the line that contains $(-1,0)$ and $(2,3)$?
   a) $y = x$
   b) $y = -x + 1$
   c) $y = 3x + 1$
   d) $y = x + \frac{1}{3}$
   e) $y = \frac{1}{3}x - 1$

2. What is the equation of the line that contains $(-1,0)$ and $(2,3)$?
   a) $y = x$
   b) $y = -x + 1$
   c) $y = 3x + 1$
   d) $y = x + \frac{1}{3}$
   e) $y = \frac{1}{3}x - 1$

\nochoices \noscrabling

These commands are used to control \emcee, the multiple-choice scrambler (see above for a complete description). They are intended for use on *Acces*’ main screen, in the Commands column. The first hides the answers to a problem, essentially making it open-ended. The second prevents the answers from being scrambled.

\nochoices and \noscrabling are related to two document-layout commands, \emceelayout and \emceeeorder, which are described in the next section. There are given here because they may be used to change the appearance of individual problems; in other words, to make local changes within a document.
To learn more about the commands `\nochoices` and `\noscrumbling`, and to see how the latter is used in problem-writing, take a look at the sample data files that come with Acces.

\begin{itemize}
\item \textbf{\fullwidth} \textbf{\halfwidth}
\end{itemize}

Normally, Acces figures out how wide each problem should be. Of course, their calculations are based on the number of columns you select, and whether you opt for answer spaces, but they do not require any intervention on your part. The purpose of `\fullwidth` and `\halfwidth` is to give you added control over the typesetting process. With those commands, you can force a problem to be printed in one column or as wide as possible.

Only one of the commands may be used in a problem. If you try to use both, the second will be ignored. The commands may appear in text or math modes; however, they must be given before the end of the first paragraph—that is, before any line breaks occur. We recommend typing `\fullwidth` or `\halfwidth` at the beginning of a problem, so there is no confusion.

Here are all of the details:

- Fullwidth problems are printed one across and as wide as possible. We say “as wide as possible” because problem numbers and answer spaces are printed as usual. The command `\fullwidth` has a visible effect on documents with more than one column. (On a one-column document, problems are already as wide as possible.) If you like, you can combine `\fullwidth` with other special effects: `\noanswer`, `\picture`, `\numberline`, etc.

- Halfwidth problems are a little more complicated. They are printed one across, no matter what layout is selected; however, they occupy about half the width of a page. We say “about half” because problem numbers and answer spaces are printed as usual. The command `\halfwidth` has a visible effect on all documents, because space to the right of the problem (at least 2 1/4 inches) is left open.

Why would you give the command `\fullwidth` or `\halfwidth`? The first is useful in many situations: when a problem contains a long expression or equation, when you want to include a large picture, or when students are supposed to give an essay-type answer. The command `\halfwidth` is not needed as often. Its main purpose is to free up space to the right of a problem. Students can use that space for a graph, proof, etc.
\linegraph \coordgraph \gridgraph

These commands produce one- and two-dimensional graphs, as shown here (the examples are $\frac{2}{3}$ actual size):

```
\linegraph
\coordgraph
\gridgraph
```

The commands can be used anywhere in a problem, in text or math mode, and they will insert a graph “on the spot” (unless they are used with \inserttright or \insertbelow). The commands do not take any parameters, but there are ways of changing the appearance of graphs (see next section).

Note: Graphs are typeset by \TeX; they are not inserted as picture files. As a result, they print quickly and very look sharp.

\numberline

This command produces a one-dimensional graph with marked intervals. It is very similar to \linegraph, except that it takes three parameters. Here is its proper usage:

```
\numberline\langle start number\rangle\langle end number\rangle\langle number of intervals\rangle
```

Let’s call the parameters, in the order they are given, $x$, $y$ and $z$. All three parameters and the value $\frac{y-x}{z}$ must be integers. If the latter condition is not met, \TeX will round off the value to the nearest integer. As always, you must use grouping symbols if $x$, $y$ or $z$ is more than a single digit.
The dimensions of a number line will vary. Its width is at least $2\frac{1}{4}$ inches (the same as \linegraph and \coordgraph), but never more than the problem which contains it. \texttt{Acces} will adjust the size of the intervals, so that everything fits in the available space; however, you should provide reasonable values to start with. In general, $x$ and $y$ should not be larger than three-digit numbers. If you need to print a lot of intervals or if $z \geq 10$, give the command \fullwidth before \numberline.

\textbf{Example}

\numberline063
\medskip % put some extra space between graphs
\fullwidth \numberline{-50}{50}{10}

results in...

\begin{figure}[h]
\centering
\input{example_numberline}
\end{figure}

\texttt{picture}

While \TeX was designed as a computerized typesetter for words, not pictures, the version used by \texttt{Acces} is quite capable of handling graphics. Currently it is able to read \texttt{PCX}, \texttt{TIF}, and \texttt{MacPaint (MCP)} files. Other formats may be supported in the future.

The command \texttt{picture} reads or inserts a graphics file. The proper usage is \texttt{picture\{\texttt{filename}\}}. The filename refers to something that you supply: a drawing, piece of clipart, or scanned image. You can use any characters for the name that DOS considers valid except left and right braces. In particular, you can use symbols like \# and \$, which are made inactive temporarily. The name must include an extension which identifies the format, like .PCX, and the file must be kept in the \texttt{Acces} subdirectory called \texttt{GRAPHICS}. For additional information, please see Appendix B.

When you use the command \texttt{picture}, the graphic is inserted on the spot. That is true whether the command is given in the middle of a sentence or between paragraphs. In fact, the graphic is placed in a box and treated like an ordinary character. Most likely, the box will be tall and lines will be spread apart to accommodate it. If the box is too big to fit in the available space, it is scaled automatically.
Note: On catalog pages, the names of graphic files are printed for reference purposes. The names are not printed on Acces’ documents. When looking at the example below, assume that everything appears on a catalog page and that the file TRIANGL7.PCX really exists (it is not included with Acces). Also, for more picture options, be sure to look at the commands \insertright and \insertbelow. They give you finer control over the placement of graphics, and their results tend to look better than the simple command \picture.

\begin{itemize}
  \item \textbf{Example}
  \end{itemize}

Solve for $x$. \RT \picture{TRIANGL7.PCX}

results in…

Solve for $x$.

\begin{center}
\begin{tikzpicture}
  \draw (0,0) -- (0,2) -- (2,0) -- cycle;
  \node at (1,1) {\texttt{TRIANGL7.PCX}};
\end{tikzpicture}
\end{center}

\begin{verbatim}
\newpic \ifnewpic
\end{verbatim}

Acces will not print the same picture twice in a row—i.e., in two consecutive problems—unless you explicitly tell them to do so. There are a couple of reasons for this. First, pictures take longer to print than text or math formulas, and generally they take up a lot of space. By not reprinting them, Acces cuts processing time and saves paper. Second, problems usually look better and are easier to read when they refer to a single picture (or refer back to a previous one).

Of course, there will be occasions when you want to reprint a picture: when students are supposed to mark it up (a geometry figure, for instance) or put an answer on top of it (a graph or table). If you want a picture to be printed in all cases, give the command \newpic. You may type it anywhere in a problem, as long as it is prior to the command which does the actual insertion.

Most of the time you should let Acces print a graphic file just once. You may think this causes difficulties when writing problems, since you do not know which ones will be selected later on. In fact, you do not even know the order in which they will be selected. But there is a simple solution. With the command \ifnewpic, you can write problems which are syntactically correct—which sound right—in every situation.
\ifnewpic is a “conditional command.” Its outcome depends on whether a condition is true or false when \TeX{} typesets a problem. The proper usage is:

\begin{verbatim}
\ifnewpic optional true text\else optional false text\fi.
\end{verbatim}

Briefly, the true text is what gets printed if a picture is new—that is, if a graphic file is actually going to be inserted. The false text is what gets printed if the picture is old—that is, if the graphic file was used previously.

When using the command, it is not necessary to type both true and false text. The ‘\texttt{\textbackslash else }’ can be skipped if there is no false text. The ‘\texttt{\textbackslash fi }’ is required; \TeX{} looks for it at the end of every conditional command. Usually the true text is something like, “Refer to the figure on the right,” and the false text, “Refer to the previous diagram.” But you can be creative and use \texttt{\ifnewpic} to convey all sorts of information. The command may be typed anywhere in a problem, as long as it comes after \texttt{\picture} (otherwise, there is no way of knowing whether the condition is true or false).

Note: The commands \texttt{\newpic} and \texttt{\ifnewpic} are difficult to demonstrate unless several related problems are typeset. To learn more about the commands, and to see how they are used in real problems, please look at the ‘Extra’ or ‘Sample’ databases that come with \textit{Acces}.

\textbf{EA} \texttt{\inserttright \inserttabove \insertbelow}

These commands alter the location of text and graphics. The first command typesets material on the right of a problem. It is used like this: \texttt{\inserttright(text or math material)}. The second and third commands typeset material above or below a problem and are used in a similar fashion.

Here are all the details:

- The text or math material can be just about anything: a picture, formula, table or list, or an entire paragraph. The material must be inside a group (between a pair of curly braces) unless it is one of these commands: \texttt{\picture}, \texttt{\linegraph}, \texttt{\coordgraph} \texttt{\gridgraph}, \texttt{\Gridgraph} or \texttt{\numberline}.

- \texttt{\inserttright} forces a problem to be “halfwidth,” in order to make room for the material on the right. \texttt{\inserttabove} and \texttt{\insertbelow} do not change any dimensions; however, they can be used in conjunction with \texttt{\fullwidth} and \texttt{\halfwidth}.
- The commands should be given at the very beginning of a problem, so that \TeX{} knows where to put everything. The only exception to this rule is when you are manually changing the width of a problem. In that case, you should type something like \texttt{\fullwidth\insertbelow...}.

- You are allowed only one insertion per problem. If you try to do another, the command is ignored.

- An insertion is treated as part of a problem. It has no effect on workspace or answer spaces, which appear as usual. If you insert a graph or number line in a problem, you may want to eliminate the answer space which appears on the far left or right. You can do this with the command \texttt{\noanswer}.

Note: The insert commands may be used for any purpose, although they were designed especially for graphics: pictures, number lines, etc. Why bother with them? The reason is that a command like \texttt{\picture}, when used alone, places the image on the spot, and this can lead to disorganized-looking pages. If you treat pictures as inserts, however, they will be lined up neatly and your pages will have a balanced look.

\textbf{Example}

\begin{verbatim}
\inserttright\picture{ABCD.PCX}
Given the figure on the right, solve for $x$ and $y$ and find the measure of each angle.

\fullwidth\noanswer\insertbelow\linegraph
Solve and graph: $2(x+4) \leq x+8 < 4-5(2-x)$
\end{verbatim}

\textit{results in...}

1. Given the figure on the right, solve for $x$ and $y$ and find the measure of each angle.

2. Solve and graph: $2(x+4) \leq x+8 < 4 - 5(2-x)$
\auxfile

This command is very similar to \picture. The most significant difference is that it reads or inputs a \TeX file instead of a graphical image. The usage is \auxfile\{⟨filename⟩\}. The filename should correspond to an ASCII file that you have created, one which contains text or commands. You can use any characters for the name that DOS considers valid except left and right braces. A file extension is optional; if you do not give one, .TEX is assumed.

When you use the command \auxfile, the contents of the file are placed in a box and inserted immediately. If you like, you can move the box by using \auxfile in conjunction with \insertright and \insertbelow, just like a picture. No assumptions are made about the box; you must control its height and width manually. Also, \TeX will not scale the box if it is too large to fit in the available space, so be sure to provide enough room in your problem, perhaps by giving the command \fullwidth.

Important: Auxiliary files must be kept in the Acces subdirectory called INPUT or in the current directory (wherever the program is run). Otherwise, \TeX will be unable to locate the file.

No example of auxfile is given here, because it serves very specialized needs. You may want to use the command if several problems share a lengthy passage, or table or chart, or any material which can be typeset and stored more efficiently on its own.

\newaux \ifnewaux

These commands are used with auxiliary files in exactly the same way that \newpic and \ifnewpic are used with pictures. The first forces \TeX to read a file twice (normally it will not). The second allows you to write conditional text, so that a problem sounds right whether or not a file is read. For more information, please refer to the corresponding picture commands.
One of the options when running Acces is to put answer spaces on the left or right side of a document. \noanswer, \subanswer and \makeanswer give you added control. With them, you can change the appearance of answers on a problem-by-problem basis. (The commands are ignored if a document is printed without answer spaces, and they have no effect at all on catalog pages.)

Here is a description of each command:

- \noanswer prevents an answer space from appearing next to a problem, regardless of which page layout is selected in Acces. The command may be given anywhere in a problem. Since end-users can print an entire document without answer spaces, you should have a good reason for eliminating the option. The command may be appropriate if students are supposed to prove something or draw a picture—or, more generally, if a problem is written in such a way that students will never need a blank line for their answers.

- \subanswer produces an extra answer space, below the one which already appears. The command can be typed anywhere in a problem, and it can be typed repeatedly. This may be appropriate if a problem has several sub-paragraphs. For each one, you could type something like \subpart\subanswer. Keep in mind, however, that each problem includes an answer space, so an additional one may not be required until you type \subpart a second time.

- \makeanswer lets you customize answer spaces on an Acces document. Normally, answer spaces appear as blank lines. With the command \makeanswer, you can substitute a graph, picture, or just about anything you like. The command is used in roughly the same manner as \inserttright and \insertbelow, although it can be typed anywhere in a problem. The proper usage is \makeanswer{text or math material}. If the text or math material is more than a single character or command, it must be put inside a group. For example, you need to type all of the braces in \makeanswer\picture{…}, but you do not need any braces in \makeanswer\linegraph.

Note: \noanswer, \subanswer and \makeanswer can easily be given at “runtime.” Thus, it is not necessary to include them in your raw data—and, in general, you should not. Also, keep in mind the order of precedence: \makeanswer overrides \subanswer, and \noanswer overrides both. If \makeanswer or \noanswer is used on Acces’ main screen, it will take precedence over commands that are written in the problem.
\fivechoices \fourchoices \TFbubbles

These commands produce “bubble answers” for multiple-choice and true-false questions. The size of the bubbles varies with font size. Here is the effect of each command with a 10 point font:

\fivechoices → a b c d e
\fourchoices → a b c d
\TFbubbles → T F

The commands can be typed anywhere in a problem and they can be used in conjunction with \makeanswer. The bubbles can also be inserted at “run-time.” For example, you could type \insertright\fivechoices or \makeanswer\TFbubbles in the commands column on Acces’ main screen.
Document layout

Note: Most of the commands in this section have no effect unless they are used at the document or outer level. For Acces, this means working outside the boundaries of a problem. There are three ways of doing this: (1) you can use the “global commands” field inside either program; (2) you can add commands to USER.TEX and then input the file; or (3) you can use \beforeproblem or \afterproblem to hold your commands until a problem ends. The first two options are explained in the Acces manual. The third is introduced here.

\ifacc

This command tells whether a document is being formatted by Acces. It is intended for portability, so that you can process your code with other programs or format files, such as Plain \TeX.

\ifacc can be used anywhere. It follows the same pattern as other conditional commands. The pattern looks like this:

\ifacc true text\else false text\fi

If there is no false text, you can leave out the command \else. The \fi at the end is mandatory.

When Acces creates a \TeX document, it always sets \ifacc to be true. If you create \TeX documents outside of Acces and typeset them with the ACC format, then \ifacc will be false. If you use another format, then you will have to be sure \ifacc is defined, in order to prevent a typesetting error.

\classname \doctitle \teacher \date

These commands store information that is entered in the Headers dialog box in Acces. Although the commands can be used anywhere, they are intended for custom headers and memos inside Acces. They do not serve any purpose at the problem-writing stage, and they are meaningless when printing catalog pages.

You can get at the information stored in \classname, \doctitle, etc., by typing the appropriate command. For example, if an Acces user enters “Mrs. Smith” in the teacher field, then the command \teacher will reproduce the name. If a certain field is left blank by a user, then the corresponding command will be “empty” (it will not produce anything, but it will not do any harm, either).
\textbf{\texttt{\textbackslash blankpics}}

This command is useful for “debugging” catalog pages or experimenting with different layouts. It greatly speeds up the previewing or printing process by hiding pictures. (Actually, it typesets an empty box where a picture would otherwise appear).

\texttt{\textbackslash blankpics} can be used anywhere, although it must be given prior to the command \texttt{\textbackslash picture} in order to have an effect. If it is typed in the “global commands” field in Acces, then all pictures will be hidden. Of course, you will have to remove the command, or comment it out, when printing your final copy.

Note: \texttt{\textbackslash blankpics} does not affect graphs or number lines. Since these things are typeset, not inserted as picture files, they do not cause much of a delay in previewing or printing.

\textbf{Example}

\[\texttt{\textbackslash blankpics \textbackslash insertright\textbackslash picture\{ABCD.PCX}\]

Given the figure on the right, solve for $x$ and $y$ and find the measure of each angle.

results in...

1. Given the figure on the right, solve for $x$ and $y$ and find the measure of each angle.

\textbf{\texttt{\textbackslash noinserts}}

This command is similar to \texttt{\textbackslash blankpics}, in that it is used mostly for debugging. But \texttt{\textbackslash noinserts} has much more of an impact: it prevents a picture, graph or anything else from being inserted into a problem; in other words, it cancels out commands like \texttt{\textbackslash picture}, \texttt{\textbackslash coordgraph} and \texttt{\textbackslash insertright}.

\texttt{\textbackslash noinserts} can be used anywhere, although it must be given prior to an insertion in order to have an effect. If it is typed in the “global commands” field in Acces, then no insertions will appear anywhere in your document. (No blank boxes, representing the insertions, will appear either.)
In addition to its use as a debugging tool, the command is good for eliminating
a picture or graph that is “hard-wired” into a problem. You can do this by
typing \noinserts in the commands column on Acces’ main screen. Since your
command will appear at the beginning of the problem, it will eliminate the
insertion which follows.

Note: When looking at the example, assume the problem is formatted by Acces
and the layout includes answer spaces on the right. Also, assume that the first
line is typed inside the Acces program; it has the effect of moving the graph from
the problem to the answer.

Example

\makeanswer\linegraph \noinserts
\insertright\linegraph
Graph the solution:
$ 2y \geq 10-\lf12y $

results in...

1. Graph the solution: $2y \geq 10 - \frac{1}{2}y$ 1. ←−−−−−−−−−−−−−−−−−−−−→

\nocatlist \noanslist \noanswerkey

These commands modify or suppress Acces’ answer key. \nocatlist eliminates
the catalog list (the record of what was picked from the database), \noanslist
eliminates the answer list, and \noanswerkey eliminates both. All three
commands are outer-level; they have no effect if typed in a problem. Also, the
commands are ignored when printing catalog pages.

Note: \noanswerkey is not a combination of the other two commands. It
suppresses the answer key altogether, whereas the others simply change the key’s
appearance.

\nonumbers

This command eliminates the problem numbers which appear on Acces’
documents or catalogs pages. The command must be used at the outer-level
to have an effect. Normally it is typed in the “global commands” field in Acces.
As an alternative, you could type something like \ifacc \nonumbers \fi in the file USER.TEX.
These commands change the width of answer spaces on Acces' documents. The first command, \medanswers, is the default; it produces answer spaces (blank lines) that are 1\$\frac{1}{4}\$ inches wide. \smallanswers reduces the width to 3\$\frac{3}{4}\$ inch, and \biganswers increases it to 2\$\frac{1}{4}\$ inches.

All three commands must be used at the outer-level. They will not have an effect if they are typed inside a problem, or if a document does not contain answer spaces. (They never have an effect on catalog pages.) Normally, if you want to change the answer width, you should do so at the beginning of a document. For example, you can type \smallanswers or \biganswers in the “global commands” field in Acces.

These commands are enormously powerful, because they let you reach outside a problem and change the appearance of a document. As you might guess, the commands take effect before and after the current problem. To keep things simple, we will limit this discussion to the second command, \afterproblem; you can easily extend the ideas to \beforeproblem.

The official usage is \afterproblem{text or commands}. If your text contains more than a single character (undoubtedly it will), or if you want to give more than one command, then you must put everything inside a group. \afterproblem will store what you type and release it later—literally after the problem.

\afterproblem is generally used in the commands column on Acces' main screen. It may also be used at the problem-writing stage, to change the layout of the document or to insert other material (e.g. a lengthy passage or a large picture). \afterproblem does not serve any purpose in the “global commands” field or in the file USER.TEX. In those places, you are already working at the outermost-level of a document.

Here are more details:

- \afterproblem has a cumulative effect. Inside a problem, it stores all of the text and commands that you provide. Thus, you can type

\afterproblem{one thing}\afterproblem{ or another}

or
\afterproblem{one thing or another}
and the result will be the same.

- Since \afterproblem accumulates what you type, there is no way to override or cancel the command if it is “hard-wired” into a problem. For this reason, you should not use the command at the problem-writing stage unless it is absolutely necessary.

- \afterproblem can be used for most of the same things as the “global commands” field in Acces. Both options let you insert something into your document, change its layout, or work directly with the typesetter. The only significant difference is that global commands take effect at the very beginning of a document.

- No safeguards are in place “after a problem,” so you should be very careful about what you do with the command. Changes you make are not automatically confined to the next problem; they will remain in effect until the end of the document or until you make further changes.

The following notes are for experienced \TeX users:

- The current problem will already have been typeset when the commands \beforeproblem and \afterproblem take effect. Therefore, you can change the position of the problem only; you cannot change its dimensions or appearance.

- \TeX may be in horizontal mode before or after a problem is placed on the page. If you want to insert some vertical mode material, you should precede it with the command \par, just to be safe.
\pageno \partno \probno

These commands keep track of the current page, part and problem number. Each has a starting value of one. Normally, there is no reason to make a change, but you can assign a different value if necessary. In that case, you would type something like \pageno=3.

All three of the commands must be used at the outer-level, otherwise they will have no effect. Also, the command \partno is relevant only when running Acces and when the option called “part numbers” is turned on.

It may be appropriate to change \pageno, \partno or \probno if one document is a continuation of another. For example, let’s say you wanted to split a review worksheet into two separate documents. The first takes up 8 pages and has 50 problems divided into 5 parts. Before printing the second document, you could type

\pageno=9 \partno=6 \probno=51

in the “global commands” field in Acces. Then the second document would pick up where the first left off.

\spacingtype \answerpos \pageshift \partnumbers

These commands change the overall appearance of a document. Each of them gets assigned a numerical value, as in \spacingtype=1. In most cases, the value must be in the range 0–2. All of the commands are relevant inside Acces, but only \pageshift and \answerpos have meaning on catalog pages (the others are ignored). As you might expect, the commands have to be used at the outer-level.

Here is a list of all values that can be assigned:

<table>
<thead>
<tr>
<th>command</th>
<th>number and meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\answerpos</td>
<td>0 = none</td>
</tr>
<tr>
<td>in Acces</td>
<td>1 = left side</td>
</tr>
<tr>
<td></td>
<td>2 = right side</td>
</tr>
<tr>
<td>\answershow</td>
<td>0 = none(hidden)</td>
</tr>
<tr>
<td>in catalogs</td>
<td>1 = paragraph style</td>
</tr>
<tr>
<td></td>
<td>2 = condensed (boxed)</td>
</tr>
<tr>
<td>\spacingtype</td>
<td>0 = condensed</td>
</tr>
<tr>
<td></td>
<td>1 = variable</td>
</tr>
<tr>
<td></td>
<td>2 = optimized</td>
</tr>
<tr>
<td>\pageshift</td>
<td>0 = centered pages</td>
</tr>
<tr>
<td></td>
<td>1 = righthand pages</td>
</tr>
<tr>
<td></td>
<td>2 = right/left pages</td>
</tr>
<tr>
<td>\partnumbers</td>
<td>0 = off</td>
</tr>
<tr>
<td></td>
<td>1 = on</td>
</tr>
</tbody>
</table>

**Summary of Commands**
In general, there is no reason to assign values to \texttt{\pageshift} and \texttt{\partnumbers}. They can be controlled through dialog boxes in \texttt{Acces}, and any change in the middle of a document would be frivolous. The commands are described here mainly for the sake of completeness; perhaps you will have need for them when processing your own \TeX\ documents.

While \texttt{\spacingtype} and \texttt{\answerpos} can also be controlled through dialog boxes, you may want to change them in the middle of a document. That is perfectly okay to do, and the method is simple: on \texttt{Acces'} main screen, go to the problem row just before you want the change to occur, then type something like

\texttt{\afterproblem{\spacingtype=0}}

in the Commands column. Answer spaces can be turned on and off in a similar manner, but keep in mind that frequent changes are disruptive.

\texttt{ \pagecolumns \textcolumns}

These commands change the number of columns in a document. The first has to do with true, vertical columns, which go \texttt{down} the page. (Normally these columns are separated by a thin rule.) The second has to do with horizontal columns or the number of problems across the page.

The commands are used in a straightforward way. You simply assign a number, as in \texttt{\textcolumns=2}. Any positive number may be assigned, but it should be kept within a reasonable range. “Page columns” should rarely be more than two and never more than four (additional columns are just too narrow to be practical). “Text columns” should be in the range 1–4 on \texttt{Acces'} documents and 1–8 on catalog pages. The commands must be given at the outer-level to have an effect.

Not all column and answer space combinations are allowed in \texttt{Acces}. If there is a conflict, the last command that is given will take precedence. For example, if you type

\texttt{\afterproblem{\answerpos=1 \textcolumns=3}}

on \texttt{Acces'} main screen, the “answer type” will be reset to 0, since there is no room for answers on a three-column document. (Answer spaces are irrelevant catalog pages, so you do have to worry about conflicts.)

Note: \texttt{\pagecolumns} can actually be assigned the value 0. If you look at a \TeX\ file created by \texttt{Acces}, you will notice it is the default. The difference between 0 and 1 is too complicated to explain here. Unless you have modified \texttt{Acces'} output routine, you can ignore the difference and safely assign either value.
\texttt{tfracstyle} \quad \texttt{PandCstyle}

These commands change certain styles on Acces’ documents. The first has to do with “text fractions,” which are formed by the command \texttt{tf}. The second has to do with permutations and combinations formed by \texttt{P} and \texttt{C}.

Each command is assigned a numerical value in the range 0–2. The meaning of each value is shown here:

<table>
<thead>
<tr>
<th>command</th>
<th>number and meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{tfracstyle}</td>
<td>0 = small</td>
</tr>
<tr>
<td></td>
<td>1 = large</td>
</tr>
<tr>
<td></td>
<td>2 = diagonal</td>
</tr>
<tr>
<td>\texttt{PandCstyle}</td>
<td>0 = nP_r</td>
</tr>
<tr>
<td></td>
<td>1 = (nP_r)</td>
</tr>
<tr>
<td></td>
<td>2 = P(n, r)</td>
</tr>
</tbody>
</table>

\texttt{tfracstyle} and \texttt{PandCstyle} can be used anywhere in a document. If they are typed in the Commands column on Acces’ main screen, they will affect just one problem. If they are typed in the “global commands” field or put in the file \texttt{USER.TEX}, they will affect the entire document. By the way, the default style in both cases is 0.

Note: The reason Acces supports different styles is that there is no agreement as to which is typographically correct. In addition, some styles are difficult to reproduce on a low-resolution printer or with a font size of 10 or 11 points. Therefore, your decision about which style to use may be based on anything from hardware requirements to personal preference.

\texttt{emceelayout} \quad \texttt{emceeeorder} \quad \texttt{emceelabel}

These commands change the appearance of multiple-choice answers—i.e., those which are bounded by the command \texttt{emcee}. Normally, there is no reason to make a change, since the style, order and labeling of answers can be controlled through dialog boxes in Acces. But there may be occasions when you want to override the program settings.

The first two commands must be assigned a numerical value in the range 0–2. For example, you can include “none of the above” as a multiple-choice answer by giving the command \texttt{emceeeorder=3}. The meaning of each value is shown here:
The last command, \texttt{emcelabel}, must be assigned a value in the range 0–8. The meaning of each is shown in the table. Notice there are alternate commands for the various styles; you may find these easier to remember or use.

<table>
<thead>
<tr>
<th>number</th>
<th>style</th>
<th>alt. command</th>
<th>number</th>
<th>style</th>
<th>alt. command</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>a)</td>
<td>\texttt{lettercount}</td>
<td>5</td>
<td>(A)</td>
<td>\texttt{lparenscount}</td>
</tr>
<tr>
<td>1</td>
<td>A)</td>
<td>\texttt{Lettercount}</td>
<td>6</td>
<td>1)</td>
<td>\texttt{numbercount}</td>
</tr>
<tr>
<td>2</td>
<td>a.</td>
<td>\texttt{lperiodcount}</td>
<td>7</td>
<td>1.</td>
<td>\texttt{nperiodcount}</td>
</tr>
<tr>
<td>3</td>
<td>A.</td>
<td>\texttt{lperiodcount}</td>
<td>8</td>
<td>(1)</td>
<td>\texttt{nparenscount}</td>
</tr>
<tr>
<td>4</td>
<td>(a)</td>
<td>\texttt{lparenscount}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Note: \texttt{subpart} makes use of the same labeling mechanism as \texttt{emcee}. As a result, you can use \texttt{emcelabel} or any of the alternate commands to change the way subparts, or sub-paragraphs, are labeled.

You can change \texttt{emcee}'s operation at any time. If you make a change in the Commands column on Acces' main screen, then just the current problem will be affected. On the other hand, if you assign a new value to \texttt{emcelayout}, etc., in the “global commands” field or in the file USER.TEX, then the change will affect your entire document.

\texttt{EA \ Lmargin \ Rmargin \ Tmargin \ Bmargin}

These commands change margins on Acces documents and catalog pages. Each gets assigned a dimension, as in \texttt{Lmargin=1.5in}. When giving a dimension, you must use a standard abbreviation for inches (in) or centimeters (cm). Decimal values are allowed. The first letter of each command, which is uppercase, tells whether you are modifying the left, right, top or bottom margin.

The margin commands must be used at an outer-level. Normally, they are given as “global commands” or they are added to the file USER.TEX. In case you are interested, the default margins in both programs are as follows:

left = .75 in  right = .75 in  top = .5 in  bottom = .6 in
\everyproblem \everydirection \everymemo

These commands serve highly specialized needs. They are modeled after the Plain \TeX commands \everyjob, \everypar, etc., which means they are assigned “token lists.” To keep things simple, we will describe just one of the commands, \everyproblem. The information can be extrapolated to \everydirection and \everymemo.

\everyproblem is intended for the file USER.TEX or the “global commands” field in Acces. The proper usage is \everyproblem={⟨text or commands⟩}. The curly braces are mandatory. The text or commands may comprise just about anything: a few words, some special instructions, a typesetting command, etc.

\everyproblem stores your text or commands for later use. In this sense, it is like \afterproblem. But there are important differences:

- The token list must be assigned at an outer-level. If you are inside a group when you type \everyproblem={⟨text or commands⟩}, then your text or commands will be essentially ignored.

- \everyproblem does not have a cumulative effect; each token list that you assign replaces the last.

- The token list is released more than once—it appears, literally, at the beginning of every problem.

While \everyproblem may seem difficult to use, it is not. In fact, it can save you a lot of time and effort when assembling a document. Here are just a few things that you can do (assume the examples are typed in the “global commands” field in Acces):

<table>
<thead>
<tr>
<th>example</th>
<th>result</th>
</tr>
</thead>
<tbody>
<tr>
<td>\everyproblem={\afterproblem\newpage}</td>
<td>starts a new page after every problem</td>
</tr>
<tr>
<td>\everyproblem={\insertright\coordgraph}</td>
<td>inserts a coordinate graph to the right of every problem</td>
</tr>
<tr>
<td>\everyproblem={Solve:\par\medskip}</td>
<td>puts directions at the beginning of every problem</td>
</tr>
</tbody>
</table>
Appendix A.
Reference tables

These tables are provided so you can quickly look up common symbols. All Greek letters and ordinary math symbols are included, as are most functions, relations, arrows and binary operators. If you have need for something that is not shown here, please see the appendix called Other Resources.

Note: The ACC format lets you use Greek letters and ordinary symbols in math or text mode. All other symbols must be used in math mode only.

**Greek letters**

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>\texttt{\alpha}</th>
<th>( \lambda )</th>
<th>\texttt{\lambda}</th>
<th>( \chi )</th>
<th>\texttt{\chi}</th>
<th>( \Delta )</th>
<th>\texttt{\Delta}</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \beta )</td>
<td>\texttt{\beta}</td>
<td>( \mu )</td>
<td>\texttt{\mu}</td>
<td>( \psi )</td>
<td>\texttt{\psi}</td>
<td>( \Theta )</td>
<td>\texttt{\Theta}</td>
</tr>
<tr>
<td>( \gamma )</td>
<td>\texttt{\gamma}</td>
<td>( \nu )</td>
<td>\texttt{\nu}</td>
<td>( \omega )</td>
<td>\texttt{\omega}</td>
<td>( \Lambda )</td>
<td>\texttt{\Lambda}</td>
</tr>
<tr>
<td>( \delta )</td>
<td>\texttt{\delta}</td>
<td>( \xi )</td>
<td>\texttt{\xi}</td>
<td>( \varepsilon )</td>
<td>\texttt{\varepsilon}</td>
<td>( \Xi )</td>
<td>\texttt{\Xi}</td>
</tr>
<tr>
<td>( \epsilon )</td>
<td>\texttt{\epsilon}</td>
<td>( \pi )</td>
<td>\texttt{\pi}</td>
<td>( \vartheta )</td>
<td>\texttt{\vartheta}</td>
<td>( \Pi )</td>
<td>\texttt{\Pi}</td>
</tr>
<tr>
<td>( \zeta )</td>
<td>\texttt{\zeta}</td>
<td>( \rho )</td>
<td>\texttt{\rho}</td>
<td>( \varpi )</td>
<td>\texttt{\varpi}</td>
<td>( \Sigma )</td>
<td>\texttt{\Sigma}</td>
</tr>
<tr>
<td>( \eta )</td>
<td>\texttt{\eta}</td>
<td>( \sigma )</td>
<td>\texttt{\sigma}</td>
<td>( \varrho )</td>
<td>\texttt{\varrho}</td>
<td>( \Upsilon )</td>
<td>\texttt{\Upsilon}</td>
</tr>
<tr>
<td>( \theta )</td>
<td>\texttt{\theta}</td>
<td>( \tau )</td>
<td>\texttt{\tau}</td>
<td>( \varsigma )</td>
<td>\texttt{\varsigma}</td>
<td>( \Phi )</td>
<td>\texttt{\Phi}</td>
</tr>
<tr>
<td>( \iota )</td>
<td>\texttt{\iota}</td>
<td>( \upsilon )</td>
<td>\texttt{\upsilon}</td>
<td>( \varphi )</td>
<td>\texttt{\varphi}</td>
<td>( \Psi )</td>
<td>\texttt{\Psi}</td>
</tr>
<tr>
<td>( \kappa )</td>
<td>\texttt{\kappa}</td>
<td>( \phi )</td>
<td>\texttt{\phi}</td>
<td>( \Gamma )</td>
<td>\texttt{\Gamma}</td>
<td>( \Omega )</td>
<td>\texttt{\Omega}</td>
</tr>
</tbody>
</table>
### Ordinary symbols

<table>
<thead>
<tr>
<th>Ordinary symbols</th>
<th>Ordinary symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>\aleph</td>
<td>\nabla</td>
</tr>
<tr>
<td>\hbar</td>
<td>\sqrt</td>
</tr>
<tr>
<td>\ell</td>
<td>\forall</td>
</tr>
<tr>
<td>\wp</td>
<td>\exists</td>
</tr>
<tr>
<td>\Re</td>
<td>\partial</td>
</tr>
<tr>
<td>\Im</td>
<td>\neg \lnot</td>
</tr>
<tr>
<td>\infty</td>
<td>\infty</td>
</tr>
<tr>
<td>\es</td>
<td>\approx</td>
</tr>
<tr>
<td>\rn</td>
<td>\QED</td>
</tr>
</tbody>
</table>

### Binary operators

<table>
<thead>
<tr>
<th>Binary operators</th>
<th>Binary operators</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>\bullet</td>
</tr>
<tr>
<td>-</td>
<td>\circ</td>
</tr>
<tr>
<td>±</td>
<td>\pm</td>
</tr>
<tr>
<td>⊕</td>
<td>\mp</td>
</tr>
<tr>
<td>\cdot</td>
<td>\cdot</td>
</tr>
<tr>
<td>\times</td>
<td>\times</td>
</tr>
<tr>
<td>÷</td>
<td>\d \div</td>
</tr>
<tr>
<td>\setminus</td>
<td>\uplus</td>
</tr>
<tr>
<td>\ast</td>
<td>\sqcap</td>
</tr>
<tr>
<td>\star</td>
<td>\sccup</td>
</tr>
</tbody>
</table>
Unary (large) operators

<table>
<thead>
<tr>
<th>∫</th>
<th>\int</th>
<th>\cap</th>
<th>\bigcap</th>
<th>\bigodot</th>
<th>\bigcup</th>
<th>\bigoplus</th>
<th>\oint</th>
</tr>
</thead>
<tbody>
<tr>
<td>\int</td>
<td>\prod</td>
<td>\cup</td>
<td>\bigcup</td>
<td>\coprod</td>
<td>\bigtimes</td>
<td>\bigsqcup</td>
<td>\ni</td>
</tr>
<tr>
<td>\sum</td>
<td>\bigvee</td>
<td>\bigtimes</td>
<td>\bigvee</td>
<td>\bigtimes</td>
<td>\bigcap</td>
<td>\bigoplus</td>
<td>\ne</td>
</tr>
</tbody>
</table>

Relations

<table>
<thead>
<tr>
<th>=</th>
<th>=</th>
<th>\sim</th>
<th>\supset</th>
<th>\subset</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;</td>
<td>&lt;</td>
<td>\simeq</td>
<td>\supseteq</td>
<td>\subseteq</td>
</tr>
<tr>
<td>&gt;</td>
<td>&gt;</td>
<td>\cong</td>
<td>\supseteq</td>
<td>\subseteq</td>
</tr>
<tr>
<td>≠</td>
<td>\neq</td>
<td>\perp</td>
<td>\subseteq</td>
<td></td>
</tr>
<tr>
<td>≤</td>
<td>\leq</td>
<td>\parallel</td>
<td>\subseteq</td>
<td></td>
</tr>
<tr>
<td>≥</td>
<td>\geq</td>
<td>\mid</td>
<td>\dashv</td>
<td></td>
</tr>
<tr>
<td>≈</td>
<td>\equiv</td>
<td>\prec</td>
<td>\vdash</td>
<td></td>
</tr>
<tr>
<td>≃</td>
<td>\succeq</td>
<td>\preceq</td>
<td>\smile</td>
<td></td>
</tr>
<tr>
<td>⩽</td>
<td>\ll</td>
<td>\propto</td>
<td>\frown</td>
<td></td>
</tr>
<tr>
<td>≡</td>
<td>\models</td>
<td>\ni</td>
<td>\bowtie</td>
<td></td>
</tr>
<tr>
<td>⊥</td>
<td>\notin</td>
<td>\models</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Functions

<table>
<thead>
<tr>
<th>\log</th>
<th>\log</th>
<th>\sin</th>
<th>\sin</th>
<th>\cot</th>
<th>\cot</th>
<th>\cosh</th>
<th>\cosh</th>
</tr>
</thead>
<tbody>
<tr>
<td>\log</td>
<td>\cos</td>
<td>\cos</td>
<td>\sin</td>
<td>\arcsin</td>
<td>\arcsin</td>
<td>\tanh</td>
<td>\tanh</td>
</tr>
<tr>
<td>\lg</td>
<td>\sin</td>
<td>\tan</td>
<td>\tan</td>
<td>\arccos</td>
<td>\arccos</td>
<td>\max</td>
<td>\max</td>
</tr>
<tr>
<td>\ln</td>
<td>\csc</td>
<td>\csc</td>
<td>\arctan</td>
<td>\arctan</td>
<td>\min</td>
<td>\min</td>
<td></td>
</tr>
<tr>
<td>\exp</td>
<td>\sec</td>
<td>\sec</td>
<td>\sinh</td>
<td>\sinh</td>
<td>\lim</td>
<td>\lim</td>
<td></td>
</tr>
<tr>
<td>\antilog</td>
<td>\models</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Arrows

| ← | \gets \leftarrow | \Leftrightarrow | \rightleftharpoons |
| ← | \Leftarrow | \rightarrow |
| → | \to \rightarrow | \Rightarrow |
| → | \Rightarrow | \longleftarrow |
| ← | \Leftarrow | \to \rightarrow |
| ← | \Leftarrow | \rightarrow |
| ← | \mapsto | \longmapsto |
| ← | \hookleftarrow | \nearrow |
| ← | \leftharpoonup | \searrow |
| ← | \leftharpoondown | \nwarrow |
| → | \rightharpoonup | \swarrow |
| → | \rightharpoonup | \swarrow |

### Delimiters

| ( | ( | ) | \rangle | \backslash | \Uparrow |
| ) | ) | ) | \langle | \| | \Vert |
| [ | [ | [ | \rceil | | | \uparrow |
| ] | ] | ] | \lceil | \vert | \Downarrow |
| \{ | \{ | \} | \brace | \rfloor | \downarrow |
| \} | \} | \} | \brace | \lfloor | \Updownarrow |
Appendix B.

Working with Graphics

Acces handles graphics in a very sophisticated way. If you want to import a picture into a problem, you simply give the command \texttt{picture} and the name of a file. (The procedure is explained in chapter 4; you should read about it, if you have not done so already.) The purpose of this discussion is to provide more details about the files: how to create them and how to store them.

Graphics are not actually made part of problems. Instead, the files are stored in their own directory and read only at the typesetting stage. This is a very efficient way of doing things. You do not waste any time cutting and pasting, you can easily import a picture into many different problems, and you never have to worry about the same picture being printed twice (this happens only if you explicitly allow it). Plus, there is only one copy of the picture taking up space on your hard disk.

While Acces provide many conveniences, neither acts as a paint or draw program. Thus, if you want to include original artwork, you will need to run a separate program. The following are known to work satisfactorily: \texttt{CorelDraw}, \texttt{Windows Draw}, \texttt{Windows Paint} and \texttt{PC Paintbrush}. In addition, you can use any number of screen capture and conversion utilities, such as \texttt{Hotshot Graphics}, or you can scan an image if you have the necessary equipment.

Here are the main requirements for graphics files:

- The format must be \texttt{PCX}, \texttt{TIF} or MacPaint (\texttt{MCP}).
- The image must have a resolution of 300 dpi.
- The width should be less than 2\(\frac{1}{4}\) inches (or 4\(\frac{1}{2}\) inches, in some cases).
- The filename must include an extension (e.g., \texttt{.PCX}).
- The file must be stored in the \texttt{GRAPHICS} subdirectory.

The format of your graphics files is very important. Perhaps you noticed that \texttt{PCX}, \texttt{TIF} and \texttt{MCP} are “bitmaps” or “raster images.” This means they are made up of individual dots or pixels, to use the official term. One advantage of bitmaps is that they can be edited in a very precise manner, literally dot by dot. Another advantage is that they are easily converted from one format to another. There are many types of bitmaps, besides those which Acces supports. Some common ones are \texttt{GIF} (used by CompuServ), \texttt{BMP} (used by MicroSoft Windows) and \texttt{PIC} (used by Lotus 123)*.

*In contrast to bitmaps, “vector images” are represented by mathematical equations. \texttt{VTEX} supports one type of vector image, known as Encapsulated PostScript or \texttt{EPS}. We discourage its use, however, because \texttt{EPS} files are not portable (they work only with PostScript printers).
There are also disadvantages to bitmaps. One is that the files can be quite large, since the color and position of each dot must be recorded. This is not a problem for Acces, because it doesn’t use multiple colors or very large images. Another disadvantage is called device dependence. The aspect ratio—the relative height and width—of a bitmap is often tied to the program which creates it, or the device which is supposed to print it. Acces gets around this last problem by requiring files to have a fixed resolution, namely 300 dots per inch. This is the highest resolution obtainable by most desktop systems: laser printers, scanners, paint programs, etc.

Although graphics must start out at 300 dpi, they will not necessarily be printed that way. Dot-matrix printers usually have resolutions of 120, 180 or 240 dpi. Acces’ typesetting program, VTEX, will automatically scale graphics to match the resolution of your printer. Nevertheless, it is necessary for all graphics files to have the same resolution. Those which do not will be measured incorrectly by VTEX.

Many paint programs give you a choice of resolution when saving an image. If you use a program which does not give you a choice, first try creating a graphics file and inserting it in a document. If it looks right—i.e., if it is scaled properly—then you do not have to do anything special. If the image gets distorted, then its resolution is something other than 300 dpi. In that case, try selecting an HP LaserJet as the printer-type inside your paint program. This trick usually forces the program to save files at 300 dpi (try it even if you don’t own a laser printer).

There are two other issues surrounding bitmap files: color and size. VTEX can handle as many as 16 different colors; however, that is quite unnecessary, because your files are going to be printed in black and white. If you are given a choice, you might as well save disk space and select the minimum number of colors, usually 2 or 4. As for size, that depends on how you go about writing problems. If you force a problem to be “fullwidth,” then you can insert a picture as wide as 4⅛ inches. If you want a picture to fit in a two-column layout, or if you want to insert a picture to the right of a problem, then its width should be no more than 2¼ inches.

Note: If a graphic is too large to fit in the available space, it will be scaled automatically. However, scaling slows down the printing process and can distort an image. For that reason, you should make an effort to create pictures of the proper size.
Appendix C.
Changes from Plain \TeX

This information is provided for users who already know a great deal about \TeX. It is meant to satisfy curiosity and to facilitate the process of importing and exporting data. In other words, if you have material that was created for Plain \TeX, this appendix will tell you how to make it compatible with EducAide’s format (called ACC).

The appendix is divided into three parts. The first part tells what features of Plain \TeX are missing, the second describes some commands which have been modified, and the third provides important information about fonts.

Deletions

The following commands or features are not in the ACC format:

- **Hyphenation tables.** There is almost no reason for them because we assign a large value to \texttt{\hyphenpenalty}. Simply put, broken words do not look good in problems and are quite distracting to students. Discretionary hyphens are still allowed; in fact, you are encouraged to use them with longer words.

- **Preloaded and unnamed fonts.** The ACC format supports all fonts that are named in Plain \TeX, such \texttt{\tenrm} and \texttt{\fivebf}. To save space, the extra font information is left out. (The truth is, scalable font technology has made “preloading” rather unnecessary.)

Modifications

Some commands are defined differently in the ACC format (for all practical purposes, there is no change in the way you use them):

- \texttt{\smallskip}, \texttt{\strut}, etc., are defined in relative instead of absolute terms—that is, by a font’s x-height instead of by points. Thus, skips and struts vary with font size.

- \texttt{\big}, \texttt{\Big}, etc., are also defined in terms of x-height, so they can be used reliably with any font. (Given a 10pt font, they produce the same size delimiters as in Plain \TeX.)

- Greek letters and ordinary math symbols can be used in math or text modes. This is a departure from Plain \TeX (but close to what is allowed in \LaTeX). If you want your data to be portable to all \TeX formats, you should use Greek letters and ordinary symbols in math mode only.
\indent and \noindent have slightly reversed roles, since the default in Acces is not to indent the first line of a paragraph. This makes \noindent somewhat superfluous, while \indent performs exactly the same role as it does in Plain \TeX.

\parindent does not control first-line indentation. It does, however, affect commands like \hang, \item, and \narrower, as you would expect. If you want change first-line indentation, you can assign a new value to \Parindent (notice the uppercase ‘P’).

< and >. These symbols are made active in math mode only. That is how Acces turns the character combinations \(\leq\) and \(\geq\) into \(\leq\) and \(\geq\).

In math mode, the ‘:’ symbol is a binary operator, not a punctuation mark. This change was made because ratios are so commonplace in secondary mathematics. For a punctuation mark, type \colon or, if you want a little extra space, \suchthat.

\TeX’s default operation is changed to “non-stop mode,” which means it will stop only if there is a fatal error or if it requires input from the user. (Of course, it will quit at the end of a job.) You can return to the operation of Plain \TeX by giving the command \errorstopmode.

Fonts

Since the ACC format uses scalable fonts, there are many topics that could be addressed here. We’ll just cover the main points:

- All two- and three-letter font commands (e.g., \bf and \cal), as well as the command \oldstyle, serve the same purpose in the ACC format as they do in Plain \TeX. This means you can obtain typewriter, slanted and smallcaps fonts by typing the commands \tt, \sl and \sc.

- All font assignments are done in groups of three. For example, the text italic font is given the names \italf@nt, \italf@@t and \italf@@@. Notice the control sequences are protected; you cannot use them unless you change the meaning of the @ symbols.

- Each group of three fonts makes up a family. The ‘f@nt’ size is the same as the user selects. The ‘f@@t’ and ‘f@@@’ sizes, which are 75% and 60% as large, are used for scriptstyle and scriptscriptstyle.

- Font families have roughly the same meaning in the ACC format as they do in Plain \TeX—that is, \fam0 is a roman font, \fam1 is math italics, etc. The numbers remain constant even when the font type is changed in Acces. For example, \fam5 refers to a slanted font, regardless of whether it is computer modern or “new math.”
Appendix D.
Other Resources

If you have trouble running Acces, find \TeX\ rather disagreeable one day, or need information about certain typesetting commands, there are various places to turn for help.

**EducAide Software**

Registered users of Acces may receive free technical support by calling 800-669-9405. If your period for toll-free support has ended, you can still call or write:

EducAide Software
Director of Customer Support
3065 Richmond Pkwy, Ste 102
Richmond, CA 94806
510-222-0600, 222-0165 fax

You may also reach us by sending email to support@educaide.com or visiting our Web site at www.educaide.com. We frequently update the site with information about Acces and database modules. We also post answers to most common questions and announce workshops and other training opportunities.

**MicroPress, Inc.**

If you want to obtain the complete \texttt{VTEX} program, with documentation, you can order it for a discounted price. Call or write:

MicroPress, Inc.
68-30 Harrow Street
Forest Hills, NY 11375
718-575-1816
TEX Users Group

Anyone who is serious about typesetting or involved in publishing technical papers is a member of this group. TUG conducts TEX workshops, sponsors a yearly meeting, and publishes journals and newsletters. To learn more about its services, contact TUG directly:

TEX Users Group
1466 NW Front Ave, Ste 3141
Portland, OR 97209
503-223-9994
TUG@mail.tug.org
Reference Books

Many books have been written about \TeX. Here are some of them:

- *The \TeXbook*, Donald E. Knuth (Addison Wesley, 1986). This is the definitive guide to \TeX, written by the program’s author. See especially chapters 4, 7, 9 and 16–19.


- *A Gentle Introduction to \TeX*, Michael Doob. This worthy and inexpensive book is available from the \TeX Users Group.

- *\TeX: Starting from 1*, Micheal Doob (Springer-Verlag, 1993). A good resource for beginning and intermediate users.

- *\TeX by Topic*, Victor Eijkhout (Addison-Wesley, 1991). This book, which is quite advanced, is a great reference for programmers.

- *Modern \TeX and Its Applications*, Michael Vulis (CRC Press, 1993). Mr. Vulis is the founder of MicroPress, Inc. His book provides a good overview of \TeX and describes what is unique to V\TeX.


- *\TeX in Practice*, Steven von Bechtolsheim (Springer-Verlag, 1993). Advertised as “the ultimate reference” to \TeX, this four-volume set is for serious users.
## Appendix E.

### Index to Commands

<p>| \about       | 86   |
| \abs         | 53   |
| \Acces       | 16   |
| \afterproblem| 79   |
| \aleph       |      |
| \alpha       | 43, 86 |
| \afterproblem|      |
| \approx      | 48, 88 |
| \arccos      | 88   |
| \arcsin      | 88   |
| \arctan      | 88   |
| \ast         | 87   |
| \asciired     |      |
| \asec        | 88   |
| \asmall        |      |
| \atilde      | 88   |
| \backslash   | 14, 89 |
| \beforeproblem| 79 |
| \big         | 92   |
| \bigcap      | 87   |
| \bigcirc     | 87   |
| \bigcup      | 87   |
| \bigodot     | 87   |
| \bigoplus    | 87   |
| \bigotimes   | 87   |
| \bigskip     | 30   |
| \bigtimes    | 87   |
| \bigtriangledown | 87 |
| \bigtriangleup | 87 |
| \biguplus    | 87   |
| \bigvee      | 87   |
| \bigwedge    | 87   |
| \blankfill   | 63   |
| \blankline   | 63   |
| \blankpics   | 77   |
| \Bmargin     | 84   |
| \bmatrix     | 60   |
| \bold        | 19, 34 |
| \bowtie      | 88   |
| \braces      | 52   |
| \bracks      | 52   |
| \bullet      | 29   |
| \bullet      | 87   |
| \C           | 54   |
| \cal         | 93   |
| \cap         | 87   |
| \cases       | 59   |
| \cdots       | 57   |
| \cdots       | 87   |
| \center      |      |
| \center       |      |
| \centered   |      |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centernoindent |    |
| \centering |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |
| \centerline |      |
| \centerline |      |
| \centerline |      |
| \centering  |      |
| \centernoindent |    |</p>
<table>
<thead>
<tr>
<th>Command</th>
<th>Page(s)</th>
</tr>
</thead>
</table>
| \
QED         | 86      |
| 
\eq         | 48, 88  |
| \quad        | 36      |
| \quadr       | 41, 86  |
| r            | 19, 51  |
| rangle       | 89      |
| ray          | 40      |
| rbrace       | 52, 89  |
| \rceil       | 89      |
| Re           | 86      |
| rfloor       | 89      |
| rho          | 86      |
| \rightarrow | 56, 89  |
| Rightarrow   | 89      |
| \rightharpoonup | 89   |
| \rightleftharpoons | 89 |
| \rm          | 18, 33  |
| \rmargin     | 84      |
| \rn          | 39, 86  |
| rneg         | 40      |
| \roundbullet | 86      |
| \rpost       | 40      |
| \rt          | 36      |
| \Rt          | 36      |
| AT           | 36      |
| \sc          | 93      |
| \searrow     | 89      |
| \sec         | 88      |
| \seconds     | 86      |
| \seg         | 40      |
| \setminus    | 87      |
| \sharp       | 86      |
| \Sigma       | 86      |
| \sigma       | 86      |
| \vartheta    | 44, 86  |
| \varepsilon  | 44, 86  |
| \varphi      | 44, 86  |
| \varpi       | 86      |
| \vartau      | 86      |
| \VS          | 86      |
| \wedge       | 87      |
| \wp          | 86      |
| \wr          | 87      |
| \x           | 47, 87  |
| \X           | 86      |
| \z           | 86      |
| \zeta        | 86      |

---

Index to Commands 101
>> (much greater) 88
\u (control-space) 18, 35
\! (negative thinspace) 15
\" (umlaut accent) 15
\# (pound symbol) 15
\$ (dollar sign) 15
\% (percent sign) 15
\& (ampersand) 15
\' (acute accent) 15
\* (mult. sign) 15
\, (thinspace) 15
\- (discretionary hyphen) 15
\ \ (thinspace) 37
\_ (underscore) 15
\^ (accent grave) 15
\{ (left brace) 52, 89
\} (right brace) 52, 89
\l (mid symbol) 89
### Appendix F.
#### Index to Topics

<table>
<thead>
<tr>
<th>Absolute Value</th>
<th>53</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACC Format</td>
<td>13, 26, 92</td>
</tr>
<tr>
<td>Access Documents</td>
<td>9</td>
</tr>
<tr>
<td>Access Program</td>
<td>26, 76, 92</td>
</tr>
<tr>
<td>Alignments</td>
<td>17, 58–61</td>
</tr>
<tr>
<td>AMSTeX</td>
<td>12</td>
</tr>
<tr>
<td>Angle Brackets</td>
<td>20</td>
</tr>
<tr>
<td>Angle Symbol</td>
<td>41</td>
</tr>
<tr>
<td>Answer Key</td>
<td>78</td>
</tr>
<tr>
<td>Answer Spaces</td>
<td>74, 81, 79</td>
</tr>
<tr>
<td>Antilog</td>
<td>53</td>
</tr>
<tr>
<td>Approximate</td>
<td>48</td>
</tr>
<tr>
<td>Arc Symbol</td>
<td>40</td>
</tr>
<tr>
<td>Argument</td>
<td>19</td>
</tr>
<tr>
<td>Arrows</td>
<td>56, 89</td>
</tr>
<tr>
<td>Aspect Ratio</td>
<td>91</td>
</tr>
<tr>
<td>Auxiliary Files</td>
<td>73, 74</td>
</tr>
<tr>
<td>Backslash</td>
<td>14</td>
</tr>
<tr>
<td>Bad Line Breaks</td>
<td>24</td>
</tr>
<tr>
<td>Bases</td>
<td>46</td>
</tr>
<tr>
<td>Binary Operators</td>
<td>87</td>
</tr>
<tr>
<td>Binomial Coefficient</td>
<td>55</td>
</tr>
<tr>
<td>Bitmaps</td>
<td>90</td>
</tr>
<tr>
<td>Blank Lines</td>
<td>62, 63</td>
</tr>
<tr>
<td>Bold Font</td>
<td>33, 34</td>
</tr>
<tr>
<td>Bold</td>
<td>18</td>
</tr>
<tr>
<td>Braces</td>
<td>52</td>
</tr>
<tr>
<td>Brackets</td>
<td>52</td>
</tr>
<tr>
<td>Bulleted Items</td>
<td>29</td>
</tr>
<tr>
<td>Carriage Return</td>
<td>18, 25, 17, 21</td>
</tr>
<tr>
<td>Case of Letters</td>
<td>15</td>
</tr>
<tr>
<td>Case Statements</td>
<td>59</td>
</tr>
<tr>
<td>Catalog List</td>
<td>78</td>
</tr>
<tr>
<td>Cents</td>
<td>39</td>
</tr>
<tr>
<td>Circle</td>
<td>41</td>
</tr>
<tr>
<td>Class Name</td>
<td>76</td>
</tr>
<tr>
<td>Column Addition</td>
<td>61</td>
</tr>
<tr>
<td>Columns, Number of</td>
<td>82</td>
</tr>
<tr>
<td>Combinations</td>
<td>54, 55, 82</td>
</tr>
<tr>
<td>Commands, Global</td>
<td>76</td>
</tr>
<tr>
<td>Comment Character</td>
<td>24, 27</td>
</tr>
<tr>
<td>Complex Fraction</td>
<td>50</td>
</tr>
<tr>
<td>Conditional Commands</td>
<td>71, 76</td>
</tr>
<tr>
<td>Conditional Text</td>
<td>7</td>
</tr>
<tr>
<td>Congruent</td>
<td>48</td>
</tr>
<tr>
<td>Continued (dots)</td>
<td>57</td>
</tr>
<tr>
<td>Control-Character</td>
<td>14</td>
</tr>
<tr>
<td>Control-Symbol</td>
<td>14</td>
</tr>
<tr>
<td>Control-Word</td>
<td>15, 20</td>
</tr>
<tr>
<td>Control-Space</td>
<td>18, 35</td>
</tr>
<tr>
<td>Coordinate Systems</td>
<td>69</td>
</tr>
<tr>
<td>Cubes</td>
<td>42</td>
</tr>
<tr>
<td>Curly Braces</td>
<td>19</td>
</tr>
<tr>
<td>Dashes</td>
<td>32</td>
</tr>
<tr>
<td>Database</td>
<td>12, 21, 7</td>
</tr>
<tr>
<td>Date (on Documents)</td>
<td>76</td>
</tr>
<tr>
<td>Debugging</td>
<td>77</td>
</tr>
<tr>
<td>Degree</td>
<td>39</td>
</tr>
<tr>
<td>Delimiters</td>
<td>52, 53, 89</td>
</tr>
<tr>
<td>Determinants</td>
<td>60</td>
</tr>
<tr>
<td>Display Fraction</td>
<td>50</td>
</tr>
<tr>
<td>Division</td>
<td>47, 50</td>
</tr>
<tr>
<td>Document Layout</td>
<td>76, 79, 81</td>
</tr>
<tr>
<td>Document Title</td>
<td>76</td>
</tr>
<tr>
<td>Dollar Sign</td>
<td>15, 22</td>
</tr>
<tr>
<td>Dots</td>
<td>57</td>
</tr>
<tr>
<td>Draw Programs</td>
<td>90</td>
</tr>
<tr>
<td>DVI Files</td>
<td>9</td>
</tr>
<tr>
<td>Dynamic Formatting</td>
<td>12</td>
</tr>
<tr>
<td>EducAide Software</td>
<td>94</td>
</tr>
<tr>
<td>Em Dash</td>
<td>32</td>
</tr>
<tr>
<td>Em Space</td>
<td>36</td>
</tr>
<tr>
<td>Emceeorder</td>
<td>67, 83</td>
</tr>
<tr>
<td>Empty Set</td>
<td>39</td>
</tr>
<tr>
<td>En Dash</td>
<td>32</td>
</tr>
<tr>
<td>Encapsulated PostScript</td>
<td>90</td>
</tr>
<tr>
<td>End-of-Line Markers</td>
<td>17, 21</td>
</tr>
<tr>
<td>Enter Key</td>
<td>21</td>
</tr>
<tr>
<td>EPS Files</td>
<td>90</td>
</tr>
<tr>
<td>Equal, Question</td>
<td>48</td>
</tr>
<tr>
<td>Equations, Systems</td>
<td>58</td>
</tr>
<tr>
<td>Error-Stop Mode</td>
<td>93</td>
</tr>
<tr>
<td>Escape Character</td>
<td>14</td>
</tr>
<tr>
<td>Exponents</td>
<td>42, 46</td>
</tr>
<tr>
<td>Extended Fraction</td>
<td>50</td>
</tr>
<tr>
<td>Feet Symbol</td>
<td>39</td>
</tr>
<tr>
<td>Files, Inputting</td>
<td>73</td>
</tr>
<tr>
<td>Fill-in-the-Blank</td>
<td>62</td>
</tr>
<tr>
<td>Fonts</td>
<td>18, 33, 92, 93</td>
</tr>
<tr>
<td>Foreign Characters</td>
<td>14</td>
</tr>
<tr>
<td>Format File</td>
<td>12, 92</td>
</tr>
<tr>
<td>Formulas</td>
<td>22</td>
</tr>
<tr>
<td>Fractions</td>
<td>20, 42, 50</td>
</tr>
<tr>
<td>Full Width</td>
<td>68</td>
</tr>
<tr>
<td>Functions</td>
<td>53, 88</td>
</tr>
<tr>
<td>Generalized Mark-up</td>
<td>12</td>
</tr>
<tr>
<td>Geometry Symbols</td>
<td>40</td>
</tr>
<tr>
<td>Global Commands</td>
<td>76, 84</td>
</tr>
<tr>
<td>Graphics Files</td>
<td>90</td>
</tr>
<tr>
<td>Graphics</td>
<td>70, 72, 90</td>
</tr>
<tr>
<td>Graphs</td>
<td>69</td>
</tr>
<tr>
<td>Greater Than or Equal</td>
<td>47, 93</td>
</tr>
<tr>
<td>Greek Letters</td>
<td>43, 44, 86, 92</td>
</tr>
<tr>
<td>Greek Variants</td>
<td>44</td>
</tr>
<tr>
<td>Grids</td>
<td>69</td>
</tr>
<tr>
<td>Grouping</td>
<td>19</td>
</tr>
<tr>
<td>Half Width</td>
<td>68</td>
</tr>
<tr>
<td>Hat</td>
<td>14</td>
</tr>
<tr>
<td>Headers</td>
<td>76</td>
</tr>
<tr>
<td>Hyphen</td>
<td>32</td>
</tr>
<tr>
<td>Hyphenation</td>
<td>92</td>
</tr>
<tr>
<td>Implicit Braces</td>
<td>19</td>
</tr>
<tr>
<td>Inches Symbol</td>
<td>39</td>
</tr>
<tr>
<td>Indentation</td>
<td>17, 22, 29</td>
</tr>
<tr>
<td>Indexed Root</td>
<td>51</td>
</tr>
<tr>
<td>Infinity</td>
<td>44</td>
</tr>
<tr>
<td>Inputting Files</td>
<td>73</td>
</tr>
<tr>
<td>Inserts</td>
<td>68–70, 70, 73</td>
</tr>
<tr>
<td>Integral Sign</td>
<td>56</td>
</tr>
<tr>
<td>Inter-Line Space</td>
<td>31</td>
</tr>
<tr>
<td>Inter-Word Space</td>
<td>17, 27, 35, 36</td>
</tr>
<tr>
<td>Intersect</td>
<td>48</td>
</tr>
<tr>
<td>Italic Font</td>
<td>33, 34</td>
</tr>
<tr>
<td>Justification</td>
<td>22</td>
</tr>
<tr>
<td>Kerning</td>
<td>8, 13</td>
</tr>
<tr>
<td>Knuth, Donald</td>
<td>6, 95</td>
</tr>
</tbody>
</table>
labels 83
\LaTeX 12
layout, document 76, 79, 81
less than or equal 47, 93
ligatures 8, 13
line breaks 7, 21, 27, 31
line spacing 31
line symbol 40
literal commands 16
logarithms 53
long division 50
lowercase letters 16
MacPaint 90
major arc 40
margins 17, 84
math formulas 46
math mode 22
math symbols 39, 86
matrix 59, 60
MCP files (MacPaint) 90
MicroPress, Inc. 94
minor arc 40
minus sign 46
mnemonic device 16
much greater 47
much less 47
multiple spaces 18
multiple-choice bubbles 75
multiple-choice 64–67, 83
multiple-part functions 59
multiplication 47
negations 49
negative sign 40, 46
new pictures 71
non-stop mode 93
not equal 49
null symbol 39
number line 69
number of columns 82
numbering pages 80
numbering parts 80, 81
numbering problems 78, 80
odd/even pages 81
ordinary symbols 86, 92
ordinary text 27
outer level 76
outer level 79
overline 49
page columns 82
page numbers 80
page offset 81
page breaks 7
paint programs 90
paragraph shapes 17
paragraph skips 30
paragraphs 21, 28
paragraphs, numbering 63, 64
parallel sign 49
parallelogram 41
parameters 18–19
parentheses 52, 53
part numbers 80, 81
PCX files 90
percent sign 24
percent (comment character) 27
permutations 54, 82
perpendicular 49
picture files 90
pictures 70–71, 77, 90–91
Plain \TeX 12, 26
Plain \TeX, changes from 92
positive sign 40
PostScript 90
problem numbers 78, 80
problem-writing 24
proportion 48
punctuation 17
quadrilateral 41
questionable equal sign 48
quote marks 33
radical sign 19, 51
raised signs 40
raster images 90
ray symbol 40
real numbers 39
reference books 95
relations 88
repeating decimals 49
right arrow 56
roman font 18, 33
roots (radicals) 51
scale graphics 91
script ℓ 44
scriptscriptstyle 93
scriptstyle 93
segment 40
sequences 57
series 57
shape of paragraphs 22
similar 48
simultaneous equations 58
spacebar 17
spaces, after commands 18
spaces, around blanks 62
spaces, in math mode 24
spacing 17, 27, 35–37
sparking, vertical 81
special effects 62
square root 51
square 45
squares 42
subparts 63
subscripts 46
summation 56
superscripts 46
symbols 86
system of equations 58
tangent 53
teacher name 76
technical support 94
\TeX 3, 6, 26, 92
\TeX Users Group (TUG) 94
text columns 82
text fractions 42, 82
text mode 22
thinspace 37
tie 24, 27
TIF files 90
tilde 14, 24, 27
times cross 47
times dot 47
token lists 84
triangle 41
trigonometry functions 53
true-false bubbles 75
TUG (\TeX Users Group) 94
typesetting 6, 8, 26
typographic details 13
underline (large) operators 87
underline 34, 49
union 48
uppercase letters 16
USER.TEX file 76, 84

104 Typesetting Guide
variable-sized delimiters 52, 53
variables 22, 24
vector graphics 90
vector 55
vertical bars 52
vertical columns 82
vertical operations 61
vertical spacing 30
VTEX program 5, 90, 94

width of answer spaces 79
width of items 68, 72
word wrap 21