

# Art of Problem Solving

## LaTeX



Want to learn how to tackle those tough MATHCOUNTS and AMC counting and probability problems? Check out Art of Problem Solving's [Introduction to Counting & Probability](#) by David Patrick.

[Login](#)  
[Bookstore](#)  
[Online Classes](#)  
[Community](#)  
[Classroom](#)  
[AoPSWiki](#)  
[LaTeX](#)  
[Resources](#)  
[Stay Informed](#)  
[About Us](#)  
[Site Guides](#)

[About](#) | [Downloads](#) | [Basics](#) | [Pictures](#) | [Reference](#) | [Help](#) | [TeXer](#)

[Layout](#) | [Symbols](#) | [Commands](#) | [Packages](#)

This page outlines some of the basics of layout in LaTeX. This is by no means an exhaustive list of what LaTeX can do, or all of what the individual commands discussed in this page can do. It's intended to be a starting point, or a quick look-up reference guide for those who have forgotten syntax or a command name.

If you would like more information, we suggest buying a book, posting on our Forum, or trying Google. Furthermore, the methods listed below are not the only way to do some of the tasks described - they're just ways we've used successfully.

[Source File Format](#)  
[Document Class](#)  
[Including Packages You'll Need](#)  
[Layout of the Page](#)  
[Starting Your Document](#)  
[Paragraphs](#)  
[Sections](#)  
[Font Sizes and Styles](#)  
[Spacing](#)  
[Justification \(Centering, etc.\)](#)  
[Tables](#)  
[Boxes](#)  
[Lists](#)  
[Referencing](#)  
[Comments](#)

Note: Rather than typing up all the examples, you can copy-paste the examples into your TeXnicCenter files. We highly recommend opening up your TeXnicCenter and trying out each of the examples as you go. It takes almost no time at all to just copy-paste, compile, and view the results.

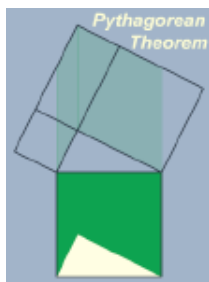
### Source File Format

The source file of a LaTeX broadly consists of two parts, the *preamble* and the document itself. The preamble consists of everything before the `\begin{document}` command. Things like margin settings, document style definitions, paragraph spacing settings, custom function definition and page numeration style are items that are set in the preamble. Often, much of the preamble is placed in a separate file and included using the `\usepackage` statement. This allows you to use the same code in many source files by just including a single line in each source file.

Our next three sections deal primarily with preamble items, while the rest cover tools you might use within your document.

[Return to top](#)

### Document Class



The first line of your source code sets the document class with the cleverly named command `\documentclass`. While LaTeX supports several classes, such as `book`, `report`, and `letter`, we will be focusing on the document class `article` in these webpages. Using `\documentclass`, we can also set the general font size for the document (which we can reset for parts of the document as required). Thus, the first line of your source code will almost always look like this:

```
\documentclass[11pt]{article}
```

If you want a slightly larger font, try `12pt`. If smaller, go for `10pt`.

Should you decide to use one of the other classes, for example if you chose to write a book with LaTeX, we suggest getting a book on LaTeX.

[Return to top](#)

## Including the Packages You'll Need

Immediately following the `\documentclass` statement we usually include all the packages we'll need using `\usepackage`. The two most common packages to include are the `amsmath` package, which defines many new mathematical symbols, and the `graphicx` package, which allows you to include images in your document. So, for instance, many of your documents may begin:

```
\documentclass[11pt]{article}
\usepackage{amsmath}
\usepackage[pdftex]{graphicx}
```

The `pdftex` in brackets is an *optional argument* to the `graphicx` package. In general, for many commands, we can add optional arguments in brackets; we'll be describing some of these as we go along.

After you have become more comfortable with LaTeX, read about how to create your own packages to simplify your source files and to allow you to easily use the same style and commands in multiple source files without having to copy over all the commands from one source file to the next.

[Return to top](#)

## Layout of the Page

In the preamble we define all the relevant settings for our page, such as margins, width of text, page sizes, etc. Here's an example of what might follow your `\documentclass` declaration to set page parameters:

```
\documentclass[11pt]{article}
\usepackage{amsmath}

\pdfpagewidth 8.5in
\pdfpageheight 11in

\setlength\topmargin{0in}
\setlength\headheight{0in}
\setlength\headsep{0in}
\setlength\textheight{7.7in}
\setlength\textwidth{6.5in}
\setlength\oddsidemargin{0in}
\setlength\evensidemargin{0in}
\setlength\parindent{0.25in}
\setlength\parskip{0.25in}
```

That may look pretty intimidating. For starters, you really only need to worry

about the first two lines if you are creating PDF documents. You can easily get away with ignoring the rest; LaTeX will simply use reasonable defaults for the other values. The two `\pdf` commands tell LaTeX what size PDF document to create. You can see that our settings above are for a standard 8.5 x 11 page.

Here are some parameters you might set (note you can use any of several different units of length, such as `in` for inches, `cm` for centimeters, or `pt` for points):

<code>\pdfpageheight,</code> <code>\pdfpagewidth</code>	Height and width of the PDF page to create (i.e. size of paper you'd print on).
<code>\topmargin</code>	Margin at top of page above all printing. Add 1 inch (so that, for example, setting <code>\topmargin</code> to <code>0.25in</code> would produce a top margin of 1.25 inches).
<code>\evensidemargin</code>	Left margin on even numbered pages. Add 1 inch (as with <code>\topmargin</code> ).
<code>\oddsidemargin</code>	Left margin on odd numbered pages. Add 1 inch (as with <code>\topmargin</code> ).
<code>\headheight</code>	Height of the header (the header is text that appears atop all pages).
<code>\headsep</code>	Distance from bottom of header to the body of text on a page.
<code>\topskip</code>	Distance from top of main text box to the baseline of the first line of text in the main text box.
<code>\textheight,</code> <code>\textwidth</code>	Height and width of main text box.
<code>\footskip</code>	Distance from bottom of body to the bottom of the footer (the footer is the text at the bottom of each page).
<code>\parskip</code>	Distance between paragraphs.
<code>\parindent</code>	Amount of indentation at the first line of a paragraph.

Many of these parameters (if not all) can be set to negative numbers. Note also that there's no explicit way to set the right margin on a page: you can control the left margin (with `\oddsidemargin` and `\evensidemargin`) and the width of the text (with `\textwidth`), which implicitly controls the right margin.

There are many other parameters that you can set in the preamble, such as the title of the document, the header style, the footer style, page numbering, etc. You can consult books or Google for more information on these areas.

[Return to top](#)

## Starting Your Document

After you've finished your preamble and are ready to start typing up your document, you start off your document with `\begin{document}`. As you might guess, you must end with `\end{document}`. Here is a complete source file that should compile correctly to create a PDF document:

```

\documentclass[11pt]{article}
\usepackage{amsmath}

\pdfpagewidth 8.5in
\pdfpageheight 11in
\setlength\topmargin{0in}
\setlength\headheight{0in}
\setlength\headsep{0in}
\setlength\textheight{7.7in}
\setlength\textwidth{6.5in}
\setlength\oddsidemargin{0in}
\setlength\evensidemargin{0in}
\setlength\headheight{77pt}
\setlength\headsep{0.25in}

\begin{document}
This is my practice document.
\end{document}

```

For the remainder of this page, all the sample code will only include items in the body of the document (i.e. the stuff between `\begin{document}` and `\end{document}`). You can just copy it into the file you created above in place of the 'This is my practice document.'

[Return to top](#)

## Paragraphs

**Reminder:** From here on, all the code is just the stuff between `\begin{document}` and `\end{document}` in your source document.

Any time LaTeX sees a blank line, it treats the next line as the start of a new paragraph. For example, try the following text:

```

Fourscore and seven years ago our fathers
brought forth on this continent a new nation,
conceived in liberty and dedicated to the
proposition that all men are created equal.

Now we are engaged in a great civil war, testing
whether that nation or any nation so conceived
and so dedicated can long endure. We are met on
a great battlefield of that war. We have come to
dedicate a portion of it as a final resting
place for those who died here that the nation
might live. This we may, in all propriety do.
But in a larger sense, we cannot dedicate, we
cannot consecrate, we cannot hallow this ground.
The brave men, living and dead who struggled
here have hallowed it far above our poor power
to add or detract. The world will little note
nor long remember what we say here, but it can
never forget what they did here.

It is rather for us the living, we here be
dedicated to the great task remaining before us--
that from these honored dead we take increased
devotion to that cause for which they here gave
the last full measure of devotion--that we here
highly resolve that these dead shall not have

```

```
died in vain, that this nation shall have a new
birth of freedom, and that government of the
people, by the people, for the people shall not
perish from the earth.
```

When you typeset this, you should find that each of the three sections above is indented and is its own paragraph. Moreover, you should see that if you don't have a full empty line between two lines in your source file, there is not only no new paragraph, but there is no line break, either. To end one paragraph and start another it is not enough to simply hit return and start typing on the next line - you must hit return twice and create an empty line for LaTeX to know to start a new paragraph.

You can also create indents wherever you want in text by adding the `\indent` command, and you can suppress the automatic indent caused by a new paragraph by using `\noindent`. For example, try typesetting this:

```
Fourscore and seven years ago our fathers
brought forth on this continent a new nation,
conceived in liberty and dedicated to the
proposition that all men are created equal.
```

```
\indent \indent Now \indent we \indent are
engaged in a great civil war, testing whether
that nation or any nation so conceived and so
dedicated can long endure. We are met on a great
battlefield of that war. We have come to
dedicate a portion of it as a final resting
place for those who died here that the nation
might live. This we may, in all propriety do.
But in a larger sense, we cannot dedicate, we
cannot consecrate, we cannot hallow this ground.
The brave men, living and dead who struggled
here have hallowed it far above our poor power
to add or detract. The world will little note
nor long remember what we say here, but it can
never forget what they did here.
```

```
\noindent It is rather for us the living, we
here be dedicated to the great task remaining
before us--that from these honored dead we take
increased devotion to that cause for which they
here gave the last full measure of devotion--
that we here highly resolve that these dead
shall not have died in vain, that this nation
shall have a new birth of freedom, and that
government of the people, by the people, for the
people shall not perish from the earth.
```

Note the effects of `\indent` and `\noindent`. Finally, understanding paragraphing rules in LaTeX is very important when using display math. Notice the difference in the following:

```
Now, if we can prove

$$\Big(2\sqrt{\frac{a}{b}}\Big)^m + \Big(2\sqrt{\frac{b}{a}}\Big)^m \geq 2^{m+1}$$

then we will be done. Dividing both sides of
this inequality by  $2^m$  yields
```

```


$$\left(\frac{a}{b}\right)^m + \left(\frac{b}{a}\right)^m \geq 2,$$

which we can verify easily by AM-GM on the reciprocals that make up the LHS, thus we have our desired

$$\left(1 + \frac{a}{b}\right)^m + \left(1 + \frac{b}{a}\right)^m \geq \left(2\sqrt{\frac{a}{b}}\right)^m + \left(2\sqrt{\frac{b}{a}}\right)^m \geq 2^{m+1}.$$


```

and

```

Now, if we can prove

$$\left(2\sqrt{\frac{a}{b}}\right)^m + \left(2\sqrt{\frac{b}{a}}\right)^m \geq 2^{m+1}$$

then we will be done. Dividing both sides of this inequality by  $2^m$  yields

$$\left(\frac{a}{b}\right)^m + \left(\frac{b}{a}\right)^m \geq 2,$$

which we can verify easily by AM-GM on the reciprocals that make up the LHS, thus we have our desired

$$\left(1 + \frac{a}{b}\right)^m + \left(1 + \frac{b}{a}\right)^m \geq \left(2\sqrt{\frac{a}{b}}\right)^m + \left(2\sqrt{\frac{b}{a}}\right)^m \geq 2^{m+1}.$$


```

In the latter, there are indents after each display math, which we clearly don't want. These are caused by the blank lines (and could be suppressed with `\noindent`).

[Return to top](#)

## Sections

For longer documents that you want to split into parts, you can use LaTeX sectioning commands. Here's an example illustrating them. If you use the `book` document class, you can also use `\chapter`, but for most documents short of books, these should be sufficient.

```

\section{In This First Section}

This is the first section.

\subsection{We Have This First Subsection}

This is the first subsection.

\subsubsection{And This Subsubsection}

A subsubsection.

\paragraph{And This Paragraph}

A notable paragraph.

\subparagraph{And This Subparagraph}

```

A notable subparagraph.

```
\subsubsection{And Then This Subsubsection}
\section{The This Second Section}
\subsection{We Have This Subsection}
```

[Return to top](#)

## Font Sizes and Styles

To change the font size, use any one of the following commands. To change it for just a portion of the page, enclose that portion in `{ }` and have the relevant font size command occur right at the beginning of the text inside the curly braces. In order from smallest to largest, the font sizes you can use are:

```
\tiny
\scriptsize
\footnotesize
\small
\normalsize
\large
\Large
\LARGE
\huge
\Huge
```

Try this out; the effects should be pretty clear:

```
When I was born, I was {\small small}. Actually,
{\scriptsize I was very small}. When I got
older, I thought some day {\Large I would be
large}, {\Huge maybe even gigantic}. But
instead, I'm not even normalsize. {\small I'm
still small.}
```

Here is a simple example that will probably show you all you need to know about bold, italics, and underlining.

```
When something is \emph{really}, \textbf{really}
important, you can \underline{underline it},
\emph{italicize it}, \textbf{bold it}. If you
\underline{\textbf{\emph{must do all three}}},
then you can nest them.
```

Here is another example that demonstrates font families:

```
You may want to write things \textsf{in a sans-
serif font}, or \texttt{in a typewriter font},
or \textsl{in a slanted font} (which is
\emph{slightly different} than italics).
Sometimes it pays \textsc{to write things in
small capitals}. You can next go to \textbf{bold
and then \textsl{bold and slanted} and then back
to just bold} again.
```

[Return to top](#)>

## Spacing

There are a few spacing items you'll find useful in LaTeX. First, you can force an extra normal-size space (as between words) by using a single backslash followed by a space. This is particularly useful after periods: LaTeX

interprets periods as ends of sentences, so it puts extra space after them, but if a period doesn't in fact end a sentence, you don't want that extra space. Try this to see an example.

```
When Mr. Rogers read this, he was confused
because the first sentence was only two words
long. Mrs.\ Rogers wasn't confused at all.
```

In math mode, it's a little different. LaTeX ignores normal spaces in math mode, so all three of the following will come out the same:

Spacing in math mode:

$$x + y$$

$$x \quad + \quad y$$

$$x+y$$

Notice that the three math expressions come out all exactly the same. In general, you can trust math mode to space things out right rather than forcing any special spacing. This means that you should write formulas in your source document to be easily readable (by you), and trust LaTeX to do the right spacing.

However, if you do need to tweak the spacing in math mode, there are some special commands:

<code>\,</code>	a small space
<code>\:</code>	a medium space
<code>\;</code>	a large space
<code>\quad</code>	a really large space
<code>\qquad</code>	a huge space
<code>\!</code>	a negative space (moves things back to the left)

Here are examples of these in action:

$$x+y$$

$$x\,y$$

$$x\;y$$

$$x\;y$$

$$x\quad y$$

$$x\qquad y$$

$$x\!y$$

Two spacing tools that can be bluntly used to move things are `\hspace` and `\vspace`.

```
I \hspace{2in} like \hspace{1in} space.
```

```
\vspace{5in}
```

```
In every direction.
```

Sometimes, `\hspace` and `\vspace` will not produce the space you want; this happens most frequently at the beginning or the end of a line or of a page. If you want to *force* space there, use `\hspace*` or `\vspace*` instead.

Two more that can be used even more bluntly are `\hfill` and `\vfill`, which cause LaTeX to create as much horizontal or vertical space as possible (within the boundaries of a page as defined in the preamble):

```
I\hfill like \hfill space.
\vfill
In every direction.
```

Finally, you can force line breaks and prevent them in LaTeX. To force a linebreak somewhere in a block of text, simply use `\\`. To prevent a linebreak, you can use `~`. The `~` in the text below prevents LaTeX from breaking a line between Professor and Reiter:

```
This website is largely a result of Professor
Harold Reiter getting me involved in math
education again.\\ Professor~Reiter's dedication
to educating eager math students is an
inspiration. Students, parents, and educators
who like this site should thank
Professor~Reiter. If you don't already know
Professor~Reiter, don't worry, you'll eventually
meet him. Professor~Reiter meets everybody.
```

Try taking out the `~` symbols, and you'll probably see Professor and Reiter get split up at least once. Notice also that the `\\` forces LaTeX to start a new line, but not a new paragraph.

[Return to top](#)

## Justification (Centering, etc.)

We can center text using

```
\begin{center} text \end{center}
```

and can justify it left or right with

```
\begin{flushleft} text \end{flushleft}
\begin{flushright} text \end{flushright}
```

```
\begin{center}
Fourscore and seven years ago our fathers
brought forth on this continent a new nation,
conceived in liberty and dedicated to the
proposition that all men are created equal.
\end{center}

\begin{flushleft}
Now we are engaged in a great civil war, testing
whether that nation or any nation so conceived
and so dedicated can long endure. We are met on
a great battlefield of that war. We have come to
dedicate a portion of it as a final resting
place for those who died here that the nation
might live. This we may, in all propriety do.
```

```

But in a larger sense, we cannot dedicate, we
cannot consecrate, we cannot hallow this ground.
The brave men, living and dead who struggled
here have hallowed it far above our poor power
to add or detract. The world will little note
nor long remember what we say here, but it can
never forget what they did here.
\end{flushleft}

```

```

\begin{flushright}
It is rather for us the living, we here be
dedicated to the great task remaining before us--
that from these honored dead we take increased
devotion to that cause for which they here gave
the last full measure of devotion--that we here
highly resolve that these dead shall not have
died in vain, that this nation shall have a new
birth of freedom, and that government of the
people, by the people, for the people shall not
perish from the earth.
\end{flushright}

```

You can also use `\raggedleft` to replace `\begin{flushright}` (and omit the `\end{flushright}`) and use `\raggedright` to replace `\begin{flushleft}` (and omit the `\end{flushleft}`)

[Return to top](#)

## Tables

The primary way to build a table is to use the `tabular` environment. Here's an example:

```

\begin{tabular}[t]{|l|ccccc|c|}
\multicolumn{7}{c}{USAMTS Scores Round
1}\hline
Name&\#1&\#2&\#3&\#4&\#5&Total\hline
John Doe&5&5&3&2&1&16\hline
Jane Doe&5&5&5&4&5&24\hline
Richard Feynman&5&5&5&5&5&25\hline
\end{tabular}

```

When you typeset that code, you should see a simple table like [this one](#). Read through the following general description of the `tabular` environment to understand how the code above produced the table.

General form of the `tabular` environment:

```

\begin{tabular}[alignment]{columns}
rows
\end{tabular}

```

The italics show where you have to put code to create your table.

*alignment* - put either `b` or `t`, or omit this completely. This determines how your table is vertically positioned with the text around it. This entry is not too important - experiment using different values (or omitting it) when you have a table in the midst of a document to get a better feel for it.

*columns* - this describes the number of columns and the alignment of each column. Put `r` for a right-justified column, `c` for a centered column, and `l` for a left-justified column. Put a `|` if you want a vertical line between columns.

For example, the column declaration `{|rr|cc|l}` will produce a table that has 2 vertical lines on the left, then two columns that are right-justified, then a vertical line, then 2 columns that are centered, then another vertical line, then a left-justified column. There are more complicating things that you can do, and even more complicated things if you include the `array` packing in your document (check a good LaTeX book for more details), but for most table, the options we've described here are sufficient.

*rows* - You can have as many rows as you like. For each row, you need an entry for each column. Each of these entries is separated by an `&`. Use `\\` to indicate that your input for that row is finished. Hence, if your column declaration was `{cccc}`, a possible row entry could be

```
5&5&5&5\\
```

If you wish for one row to have fewer columns (i.e. one column takes up several of the usual table columns), use the command `\multicolumn`. In the example above, we had as our first row

```
\multicolumn{7}{c}{USAMTS Scores Round 1}\\
```

The first `{ }` indicates how many regular columns this entry will take up. The second `{ }` indicates whether the text in this entry is right (`r`), left (`l`) or center (`c`) justified. The final `{ }` contains our entry. As with the regular column declaration, use `|` if you want a vertical line before or after the entry of `\multicolumn`.

In general, you can use `\vline` to introduce a vertical line anywhere in a table (try putting one between `John` and `Nash` in the example below and see what happens).

Finally, at the end of some of the rows in our example, we have the command `\hline`. This produces a horizontal line after the row it follows. If you want a horizontal line atop a table, use `\hline` right before the first row. If you only want a horizontal line under a portion of the row, use `\cline{start column-end column}` as indicated in the example below:

```
\begin{tabular}[t]{|l|l|cccc|c|}\hline
\multicolumn{7}{|c|}{USAMTS Final Scores by
Round}\\hline
Medal&Name&\#1&\#2&\#3&\#4&Total\\\hline\hline
&Richard Feynman&25&25&25&25&100\\\cline{3-7}
Gold&Albert Einstein&25&25&25&25&100\\\cline{3-7}
&Marie Curie&25&24&24&25&98\\\hline
Silver&John Nash&20&20&25&24&89\\\hline
&Jane
Doe&23&\multicolumn{2}{c}{None}&25&48\\\cline{3-7}
None&John
Doe&\multicolumn{2}{c}{None}&25&20&45\\\cline{3-7}
&Lazy
Person&5&\multicolumn{3}{c|}{None}&5\\\hline
\end{tabular}
```

When you typeset this, you should get output [like this](#). Note how we made a double horizontal line after the table headings.

Finally, sometimes you'll want to create a table that consists solely of items

in math mode. For such a table, use the `array` environment. The `array` environment works exactly like `tabular`, except that all its entries are rendered in math mode:

```

\begin{array}[b]{ccc}
x&y&z\\
y&x&z\\
1&2&3
\end{array}

```

Change both `array` declarations to `tabular` and delete the `$$`'s and see what happens. You can do a number of other things with the `array` and `tabular` environments, but the above should cover most of what you'll want to do with them.

If you build a table in which some entries are text that will take up multiple lines, you'll probably want to learn about [boxes](#).

[Return to top](#)

## Boxes

To create boxes of text that behave differently from the rest of the text, we can use

```
\makebox[width][pos]{text}
```

The *width* sets the width of the box. The *pos* sets the positioning of the text - either *r* (right justified text), *l* (left justified), or *s* (stretched to fill the box). If the *pos* parameter is left out, as in `\makebox[1in]{centerme}`, the text is centered. The *text* is placed in the box. If you want to draw a box around the text, use `\framebox` just as you would use `\makebox`.

`\mbox{text}` and `\fbox{text}` are quick versions of `\makebox` and `\framebox`, which create a box to fit the size of the text.

Try the following example to see some of these features:

```

I can create basic boxes for text
\makebox[2in]{like this}. Notice that there's a
2in wide space with `like this' in the middle of
it.

If I want to put a box around the text, I can
use a frame box. The result looks
\framebox[2in]{like this}.

I can also justify the text to the right within
a box \makebox[1.5in][r]{like so} or
\framebox[2.5in][l]{like so}.

We can also use quick versions of these. We can
just \mbox{do this} or \fbox{this} to create a
quick box that's exactly the size of what we put
in it.

```

In some of the boxes above, you'll see the limitations of these boxes. In particular, LaTeX won't do a line break in the middle of a box, so your box might run off the right side of the page. For boxes that span many lines, we can use

```
\parbox[pos]{width}{text}
```

The *pos* is used to align the box with the text outside the box - set it to either

`b` or `t` to align the bottom or top edge of box with the current baseline (try both and see what happens). You can also just omit this parameter and the box is centered vertically on the current line of text. Look at the following example to see how `pos` affects the box.

The parameter `width` sets the width of the box and the `text` is what is in the box.

Here's an example:

```
\parbox[b]{2in}{I like using parbox to create
funny little boxes of text all over my page.
This one has its bottom edge aligned with the
current line}
CURRENT LINE.
\parbox[t]{2in}{The top of my text is aligned
with that current line.}

\parbox{1.5in}{I'm just centered on the current
line.} CURRENT LINE \parbox{2.5in}{You probably
got a few Overfull or Underfull warnings when
you typeset this. Sometimes narrow boxes will do
that; if you're happy with the output, don't
sweat it.}
```

`\parbox` is very useful with tables, as it can be used to include multirow input on a single row. Without it, long input in a cell will sometimes cause a table to stretch out into the margins. Note how boxes are used to create the tables below, and notice the effect of not using the `\parbox` in the second table.

```
\begin{tabular}[t]{ccccc}
\makebox[2.0in]{}&&\makebox[1.5in]{}&&
\makebox[1.5in]{}\\ \cline{1-1}\cline{3-
3}\cline{5-5}
Student Name&&\parbox[t][0.2in][1.5in]{Art of
Problem Solving Community Username}&&Student
Email\\
\end{tabular}

\begin{tabular}[t]{ccccc}
\makebox[2.0in]{}&&\makebox[1.5in]{}&&
\makebox[1.5in]{}\\ \cline{1-1}\cline{3-
3}\cline{5-5}
Student Name&&Art of Problem Solving Community
Username&&Student Email\\
\end{tabular}
```

[Return to top](#)

## Lists

There are three simple versions of lists you can make with LaTeX.

To separate list items with bullet points, use `itemize`:

```
\noindent Here's my list:

\begin{itemize}
\item Item 1.
\item Item 2.
\item Item 3.
```

```
\end{itemize}
```

The `\noindent` is there to exhibit how the list is indented.

If you want a numbered list, use `enumerate`:

```
\noindent Here's my list:

\begin{enumerate}
\item Item 1.
\item Item 2.
\item Item 3.
\end{enumerate}
```

If you want to make your own headings for each item, use `description`:

```
\noindent Here's my list:

\begin{description}
\item[Heading 1.] Item 1.
\item[Heading 2.] Item 2.
\item[Heading 3.] Item 3.
\end{description}
```

Lists can be nested up to four levels. Here's one nested 2 levels deep:

```
\noindent Here's my list:

\begin{itemize}
\item Item 1.
\begin{itemize}
\item List 2, Item 1
\item List 2, Item 2
\end{itemize}
\item Item 2.
\item Item 3.
\end{itemize}
```

You can use the command `\renewcommand` to change the labels for items. The labels for `itemize` have the names `\labelitemi`, `\labelitemii`, `\labelitemiii`, `\labelitemiv` (one for each level of nesting, with `\labelitemi` being the first):

```
\renewcommand{\labelitemi}{\heartsuit}
\renewcommand{\labelitemii}{\spadesuit}

\noindent Here's my list:

\begin{itemize}
\item Item 1.
\begin{itemize}
\item List 2, Item 1
\item List 2, Item 2
\end{itemize}
\item Item 2.
\item Item 3.
\end{itemize}
```

As expected, you can also get pretty fancy with `enumerate`:

```
\renewcommand{\labelenumi}{\Roman{enumi}.}
```

```

\renewcommand{\labelenumii}{\Roman{enumi}}.
\alph{enumii}}

\noindent Here's my list:

\begin{enumerate}
\item Item 1.
\begin{enumerate}
\item List 2, Item 1
\item List 2, Item 2
\end{enumerate}
\item Item 2.
\item Item 3.
\end{enumerate}

```

The `\labelenumi` and `\labelenumii` are our labels for the two levels of nesting (as with `itemize`, there are four levels possible). The

```
\renewcommand{\labelenumi}{\Roman{enumi}}.
```

causes each item in the primary list to have a capitalized Roman numeral followed by a period. The `enumi` is the counter for the main items. The

```
\renewcommand{\labelenumii}{\Roman{enumi}}.
\alph{enumii}}
```

causes each item in the secondary list to have a capitalized Roman numeral of the primary list followed by a period, then a lowercase letter corresponding to the item of the secondary list. As you probably guessed, `enumii` is the counter that tells us which number of the secondary list we're on.

In addition to `\Roman` and `\alph`, you can use `\arabic`, `\roman`, and `\Alph` to format counter numbers such as `enumii` to numbers, lowercase Roman numerals, or uppercase letters. Try them in place of `\Roman` and `\alph` above.

If you're still looking for even fancier ways to make lists, try looking [here](#) (this links to a mirror of a LaTeX manual).

[Return to top](#)

## Referencing

Sometimes you'll want to refer to past equations, or even to past pages. Rather than having to manually number equations then change your text if the equation labels change, or having to manually put in page numbers which might change later, you can use the referencing built into LaTeX. To tell LaTeX to note where something is so you can refer to it later, use

```
\label{label name}
```

To refer to it later, use

```
\ref{label name}
```

or

```
\pageref{label name}
```

The latter gives you the page number, and the former gives you the Theorem number or equation number or whatever other special number might be attached to the item labeled with `label name`.

When using `\label`, you'll have to compile your LaTeX code **twice**. The first compilation will produce a `.aux` file that contains all the label information based on your `\label` commands. The second compilation reads and uses that information to fill out your `\ref` and `\pageref` commands.

Here's an example with several uses of `\label` and `\ref`.

```
\section{Some Sums}

Here are a few sums I know.\label{sec:formulas}

\begin{eqnarray}
1+2+3+\cdots+n&=&\frac{n(n+1)}{2}\label{linear} \\
1^2+2^2+3^2+\cdots+n^2&=&\frac{n(n+1)(2n+1)}{6}\label{squares} \\
1^3+2^3+3^3+\cdots+n^3&=&\frac{n^2(n+1)^2}{4}\label{cubes}
\end{eqnarray}

I can find the sum of the first 10 squares
easily with formula~(\ref{squares}) above.

\pagebreak

\section{A Cool Relationship}

Take a look at formulas~(\ref{linear})
and~(\ref{cubes}) on page~\pageref{linear} of
section~\ref{sec:formulas}. Notice that the
right side of~(\ref{cubes}) is the square of the
right side of~(\ref{linear}).
```

Remember, you have to compile the previous example **twice** in order to see the references. After the first compilation, LaTeX will warn you about "undefined references", and you'll see "???" in your document where the references should be. Compile it a second time, and the references will work.

The `\label` commands in the `\begin{eqnarray}` allow us to use `\ref` later to refer back to the equations. The `\label{sec:formulas}` allows us to use `\ref` to refer to the section. Note that we can use the same label for the `\pageref` as we use for the `\ref`. Also, note the `~` before each of the references; these aren't required, but are used for style concerns. They prevent a line break right before the reference. Generally, it's poor style to start off lines with numbers or references; these `~` symbols prevent that if possible.

[Return to top](#)

## Comments

You can leave yourself (or later users of your source files) notes by using `%`. Anything in a line after `%` is ignored when the file is compiled. Try this:

```
%You won't see this in the final document.
You do see this.
```

You'll see that comments are grayed out in TeXnicCenter. Comments are very useful to guide other readers of your source file, and to guide you in case you'll come back to a file later. You'll find comments throughout our sample files.

[Return to top](#)



Looking for a challenging geometry text? Preparing for MATHCOUNTS or the AMC exams? Check out Art of Problem Solving's [Introduction to Geometry](#) by Richard Rusczyk.

© Copyright 2007 AoPS Incorporated. All Rights Reserved. • Foundation • Privacy • Contact Us