TEXnique #1

Starting Out:

Every LaTeX document must have three commands in it. A Documentclass command and a begin and end document commands.

```
\documentclass[12pt]{article}
\begin{document}

\end{document}
```

The documentclass command has two main parts. The portion between the square brackets are the options, which may be omitted. The portion between the curly brackets is the type of document to be created. Most of the time we will be creating articles. The option of 12pt makes the font size of the normal text 12 points high. This is just like setting a Word font size to 12. Descriptions of the document styles and the options are given on pages 8–9 of the on-line manual.

The area between the documentclass command and the begin document command is called the preamble. Commands that change the entire document appearance tend to go here, like margin settings, page numbering and package loading. We will do more with these types of commands in the near future.

Text:

Text in LaTeX is easy. You simply write the text in the editor and when you compose the document LaTeX automatically reformats the paragraphs and lines so that they look good. It even hyphenates for you automatically.

The only thing you need to remember is that extra spaces are removed and to begin a new paragraph you simply need to insert a blank line, LaTeX will do the indenting for you.

Like this.

The above text was produced from the following text.

```
Text in \LaTeX{} is easy.  You simply write the text in the editor
and when
you
compose the document \LaTeX{} automatically reformats the paragraphs
and lines so that they look good.
It even hyphenates
for you automatically.

The only thing you need to remember     is that
extra spaces      are      removed and to begin a new paragraph you
simply need to insert a blank line, \LaTeX{} will do the indenting
for you.


Like this.
```

As far as fonts and font sizes go you don't have a lot of options. Frankly, you shouldn't. LaTeX was built for doing mathematics and you really don't need 500 different fonts to write a mathematics article. The options you do have for font sizes and styles is given on page 65 of the on-line manual.

Delimiters:

A delimiter is a set of curly brackets, { and }. What ever is done inside the brackets does not affect the outside. For example,

```
This is a {\bf bold} word.
```

produces

This is a **bold** word.

and

```
This is a \bf bold word.
```

produces

This is a **bold word.**

Spacing:

There are severalcommands that allow you to add vertical and horizontal spacing. Of these, the ones I use the most are `\vskip`, `\vspace` and `\hspace`.

The syntax for the `\vskip` is `\vskip` *length* where the length is given as a number of standard units. Units used by LaTeX are given on page 69 of the on-line manual. For example,

```
Line 1.
\vskip 12pt
Line 2.
```

produces

Line 1.

Line 2.

The syntax for `\vspace` and `\hspace` are similar, `\vspace`{*length*} and `\hspace`{*length*}, where the length is given as a number of standard units. For example,

```
Line 1.

\vspace{12pt}

Line 2.
```

produces

Line 1.

Line 2.

and

```
Line 1. \hspace{50pt} Line2.
```

produces

Line 1. Line 2.

Note that the `\vspace` command should only be used between paragraphs and thus should have blank lines before and after the command. If you need to insert space between lines that do not start new paragraphs you should use `\vskip`.

More information about spacing can be found in the on-line manual.

TEXnique #2

A Little Mathematics:

The whole idea in taking the time to learn LaTeX is so that mathematics is easier and looks better. At this point it may appear that typing mathematics is not worth the bother of learning this system but believe me, it is. Okay, there are two types of mathematical typing in LaTeX, in-line and display. The in-line looks like $f(x) = 3x^2 - 2x + 4$ and the displayed looks like

$$\iint\limits_R \frac{xy}{\sqrt{x^2+y^2+1}}\, dA$$

To create in-line mathematics simply place the mathematical "code" between two dollar signs (one on each side). For example, the in-line code for the quadratic function above was `$f(x) = 3x^2-2x+4$`. To create displayed mathematics you simply put the mathematical "code" between `\[` and `\]`. For example, the above double integral was done by

```
\[
\iint\limits_R \;\; \frac{xy}{\sqrt{x^2+y^2+1}} \; dA
\]
```

The syntax for the mathematical code is what the majority of the manual discusses. We will go over a few of the things you can do but it will be far from an exhaustive treatment.

The very first thing we need to do is load in some symbols that may be needed. If we start the document with three package loading commands (shown below) we will load in all of the symbols that are given in the tables in the LaTeX manual.

```
\documentclass[12pt]{article}
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{amsmath}
\begin{document}
```

Remember that the section between the `\documentclass` and the `\begin{document}` is called the preamble. The command placed in the preamble are usually those that will alter the entire document. You may have noticed that the standard margins are a bit wide. This is because the printed portion is the correct size for many mathematical journal publications. There are a number of ways to change the margins in a LaTeX document. I tend to use just one method that takes four commands.

```
\documentclass[12pt]{article}
\voffset=-1in
\hoffset=-.65in
```

```
\textheight=9in
\textwidth=6.5in
\usepackage{amsfonts}
\usepackage{amssymb}
\usepackage{amsmath}
\begin{document}
```

The `\voffset=-1in` is a vertical offset. A positive vertical offset moves the text down the page and a negative vertical offset moves the text up the page. The `\hoffset=-.65in` is a horizontal offset. A positive horizontal offset moves the text to the right and a negative horizontal offset moves the text to the left. So these two commands will move the upper left corner of the text area further up and to the left. The `\textheight=9in` sets the vertical height of the text area and `\textwidth=6.5in` sets the width of the text area. This document was set with these margins. One thing to note about shifting the text area is that different previewers place the text area in different places. So if you set these offsets and widths to look good when you pdf a document using the system I gave you, you may need to reset them if you use the previewer provided on the Linux computers in 105.

I am clearly digressing here. Let's get back to mathematics. As you saw in the last LaTeX exercise, you do exponentiation using the caret key, `^`. You can also do subscripting by using the underscore key, `_`. For example, `$x_1$` gives x_1 and `$x^3_1$` gives x_1^3. You can also do subscripts on subscripts and subscripts on superscripts as long as you use the curly brackets to group the expression in the correct way. For example, `$x_{a_1}$` gives x_{a_1}, `$x^{b^{r^2}_s}_{a^3_t}$` gives $x^{b^{r^2}_s}_{a^3_t}$ and `$x^{x^{x^{x^x}}}$` gives $x^{x^{x^{x^x}}}$.

Fractions are done with the `frac` command. For example, `$\frac{x-1}{x^2+1}$` gives $\frac{x-1}{x^2+1}$. Fractions almost always look better in display mode. The above in display mode looks like,

$$\frac{x-1}{x^2+1}$$

We can, of course, do fractions inside fractons as well. For example,

```
\[ \frac{x-\frac{1}{x}}{\frac{x^2}{x-7}+1} \]
```

gives

$$\frac{x-\frac{1}{x}}{\frac{x^2}{x-7}+1}$$

Since the syntax is the same for in-line mode as it is for display mode we will drop the mathematical mode delimiters.

Roots also have an easy syntax, it is just the `sqrt` command. This command has two forms, `\sqrt{x-1}` gives $\sqrt{x-1}$ and `\sqrt[7]{x-1}` gives $\sqrt[7]{x-1}$. We can combine all of these mathematical codes at will. For example,

```
\frac{x-\sqrt{\frac{1}{\sqrt{\sqrt[3]{x}}}}}{\frac{x^2}{\sqrt[5]{x-7}}+1}
```

gives

$$\frac{x-\sqrt{\frac{1}{\sqrt{\sqrt[3]{x}}}}}{\frac{x^2}{\sqrt[5]{x-7}}+1}$$

Special functions, like sine and cosine, can be created simply by putting a backslash in front of the standard function. For example, `\sin(x)` gives $\sin(x)$. Note what happens if we do not use the backslash `sin(x)` gives $sin(x)$. The function name comes out in italics when the backslash is not used, this is not accepted as good typesetting. A list of supported mathematical functions is on page 39 of the on-line manual.

Greek letters can be created using the backslash in front of the name of the letter. For example, `\alpha` gives α and `\psi` gives ψ. Page 47 of the on-line manual gives tables of lower and upper case Greek letters. Also note that any of the symbols in the charts on pages 47 – 53 are simply produced by using the command beside the symbol. We will not go over these but concentrate on LaTeX structures.

Parentheses have a little trick to them. If you type in `(\frac{1}{x})^5` in display mode you get

$$(\frac{1}{x})^5$$

which is a bit on the ugly side. To make the parentheses big enough for what they surround you need the `left` and `right` commands. For example, `\left( \frac{1}{x} \right)^5` in display mode gives

$$\left(\frac{1}{x}\right)^5$$

These commands can be nested. For example,

```
\left( \frac{\left( \frac{1}{\sqrt{x^2+1}} \right)^5}{\frac{x^3-
\frac{2}{x-1}}{x+7}} \right)^{\left(\frac{3}{4}\right)^2}
```

gives

$$\left(\frac{\left(\frac{1}{\sqrt{x^2+1}}\right)^5}{\frac{x^3-\frac{2}{x-1}}{x+7}}\right)^{\left(\frac{3}{4}\right)^2}$$

TEXnique #3

Some More Mathematics:

In the last TEXnique we saw how to do in-line and displayed mathematics. The formulas we looked at in this handout were restricted to a single line. Many times we want to place several lines in a formula that are lined up in some nice way. There are two environments that do this, the eqnarray and the eqnarray*. These both line up the the equal signs (or middle symbol) in all of the expressions. The difference between the eqnarray and the eqnarray* environments is that the eqnarray will place numbers beside the equations, usually for reference. For example,

```
\begin{eqnarray*}
\iint\limits_R \;\; xye^x \; dA & = & \int_0^1 \int_0^2  xye^x \; dy \; dx \\
 & = & \int_0^1  xe^x \left[ \frac{1}{2} y^2 \right]_0^2  \; dx \\
 & = & 2 \int_0^1  xe^x \; dx \\
 & = & 2 \left[ xe^x-e^x \right]_0^1  \\
 & = & 2 [-(-1)] \\
 & = & 2
\end{eqnarray*}
```

produces

$$\begin{aligned}
\iint\limits_R \; xye^x \; dA &= \int_0^1 \int_0^2 xye^x \; dy \; dx \\
&= \int_0^1 xe^x \left[\frac{1}{2} y^2 \right]_0^2 \; dx \\
&= 2 \int_0^1 xe^x \; dx \\
&= 2 \left[xe^x - e^x \right]_0^1 \\
&= 2[-(-1)] \\
&= 2
\end{aligned}$$

and

```
\begin{eqnarray}
\iint\limits_R \;\; xye^x \; dA & = & \int_0^1 \int_0^2  xye^x \; dy \; dx \\
 & = & \int_0^1  xe^x \left[ \frac{1}{2} y^2 \right]_0^2  \; dx \\
 & = & 2 \int_0^1  xe^x \; dx \\
 & = & 2 \left[ xe^x-e^x \right]_0^1  \\
 & = & 2 [-(-1)] \\
 & = & 2
\end{eqnarray}
```

produces

$$\begin{aligned}
\iint_R \; xye^x \; dA &= \int_0^1 \int_0^2 xye^x \; dy \; dx && (1)\\
&= \int_0^1 xe^x \left[\frac{1}{2}y^2\right]_0^2 \; dx && (2)\\
&= 2\int_0^1 xe^x \; dx && (3)\\
&= 2\left[xe^x - e^x\right]_0^1 && (4)\\
&= 2[-(-1)] && (5)\\
&= 2 && (6)
\end{aligned}$$

When using the eqnarray environment one seldom needs to number each expression. We can suppress the numbering by using the `\nonumber` command. For example,

```
\begin{eqnarray}
\iint\limits_R \;\; xyeˆx \; dA & = & \int_0ˆ1 \int_0ˆ2  xyeˆx \; dy \; dx \\
 & = & \int_0ˆ1  xeˆx \left[ \frac{1}{2} yˆ2 \right]_0ˆ2  \; dx \nonumber \\
 & = & 2 \int_0ˆ1  xeˆx \; dx  \\
 & = & 2 \left[ xeˆx-eˆx \right]_0ˆ1 \nonumber \\
 & = & 2 [-(-1)] \nonumber \\
 & = & 2 \nonumber
\end{eqnarray}
```

produces

$$\begin{aligned}
\iint_R \; xye^x \; dA &= \int_0^1 \int_0^2 xye^x \; dy \; dx && (7)\\
&= \int_0^1 xe^x \left[\frac{1}{2}y^2\right]_0^2 \; dx \\
&= 2\int_0^1 xe^x \; dx && (8)\\
&= 2\left[xe^x - e^x\right]_0^1 \\
&= 2[-(-1)] \\
&= 2
\end{aligned}$$

In general, the eqnarray and eqnarray* environments start out with `\begin{eqnarray}` or `\begin{eqnarray*}` then we have expressions of the form

expression `&` *relation* `&` *experssion* `\\`

The `&` separates the experssions and the relation can really be anything but it is usually some reation between the expressions. The end of each line (except the last) should be the line

break character \\. When the array is finished the environment ends with \end{eqnarray} or \end{eqnarray*}.

Another thing we have not looked at yet are mathematical expressions that have "limits". For example, the limit usually has something like $x \to a$ underneith it. Sums, products and integrals also have limits or bounds. Let's look at some of these. In general, to create one of these you type the command for the main symbol, then use the underscore _ before the lower limit and the ^ before the upper limit. For example,

```
\[
\sum_{i = 0}^\infty \frac{1}{2^i} = 1
\]
```

produces

$$\sum_{i=0}^{\infty} \frac{1}{2^i} = 1$$

```
\[
\lim_{\theta \rightarrow 0} \frac{\sin(\theta)}{\theta} = 1
\]
```

produces

$$\lim_{\theta \rightarrow 0} \frac{\sin(\theta)}{\theta} = 1$$

```
\[
\prod_{i = 0}^\infty a_i = t
\]
```

produces

$$\prod_{i=0}^{\infty} a_i = t$$

```
\[
\int_0^1  xe^x \; dx
\]
```

produces

$$\int_0^1 xe^x \; dx$$

```
\[
\bigoplus_{\lambda \in \Lambda} \; \mathfrak{A}_\lambda
\]
```

produces

$$\bigoplus_{\lambda \in \Lambda} \mathfrak{A}_\lambda$$

and

```
\[
\bigcap_{\lambda \in \Lambda} \; U_\lambda = \emptyset
\]
```

produces

$$\bigcap_{\lambda \in \Lambda} U_\lambda = \emptyset$$

These expressions also work with in-line expressions except that the limits or bounds are usually written beside the main symbol instead of on the top and bottom. For example, `\cap_{\lambda \in \Lambda} U_\lambda = \emptyset` produces $\cap_{\lambda \in \Lambda} U_\lambda = \emptyset$ and `\int_0^1 xe^x \; dx` produces $\int_0^1 xe^x \; dx$.

Let's take a quick look at arrays. The array is simply an environment that must be in math mode which produces a rectangular array of symbols. It is usually used for matrices but is certainly not limited to matrices. Let's look at an example before we discuss the general syntax.

```
\[
\begin{array}{lclr}
3 & 5 & -1 & 10 \\
a & \sqrt{2} & \frac{1}{2} & 0 \\
1 & 3 & 5 & 7
\end{array}
\]
```

produces

$$\begin{array}{lclr}
3 & 5 & -1 & 10 \\
a & \sqrt{2} & \frac{1}{2} & 0 \\
1 & 3 & 5 & 7
\end{array}$$

The general syntax for the array environment begins as every environment with the begin statement and the type of environment, in this case an array, `\begin{array}`. The next portion `{lclr}`, indicates the wat that the entries are justified in their column. There is one letter for each column and the letter can be either a l, c or r. The l stands for left justified, c for center and r for right. So in our above example the first and third columns are left justified, the second is centered and the last is right justified. You can have as many columns as you wish, just make sure that the number of letters in the alignment portion matches the number of columns. The entries of the array are separated with `&` and there is the line break character `\\` at the end of each line except the last.

Note that the array does not have any brackets on it as a matrix would. You can add them using the left and right commands used in the last TEXnique. For example,

```
\[
\left[
\begin{array}{ccc}
3 & 5 & 1 \\
1 & 7 & 8 \\
1 & 3 & 5
\end{array}
\right]
\]
```

produces

$$\left[\begin{array}{ccc} 3 & 5 & 1 \\ 1 & 7 & 8 \\ 1 & 3 & 5 \end{array}\right]$$

Arrays can also be used for any mathematical type setting that needs to be placed in some type of grid. For example,

```
\[
\begin{array}{ccc}
A & \longrightarrow & C \\
\downarrow &  & \downarrow \\
B & \longrightarrow & D
\end{array}
\]
```

produces

$$\begin{array}{ccc} A & \longrightarrow & C \\ \downarrow & & \downarrow \\ B & \longrightarrow & D \end{array}$$

More Help:

There is another help system on LaTeX you may wish to use. When you loaded the MiKTeX system there were several help systems that were loaded as well. To get to these help systems go to Start ≻ MiKTeX ≻ Help. The LaTeX2e Reference is particularly good if you want to find a command or syntax.

TEXnique #4

Environments:

An environment in LATEX is a way to make specialized paragraphs. That is, you can make paragraphs that hold mathematics like arrays or equation arrays which you saw in the last TEXnique or you can make paragraphs that hold quotations and even poetry although I have no idea why someone would write poetry in LATEX. A list of environments can be found on the LATEX2e help system. They can also be found throughout the on-line LATEX manual but a short description starts on page 26. For a listing you should go to page 82 of the index. We will not look at all of the environments available in LATEX but we will look those that will probably be the most useful to you. First of all environments all have the same syntax,

```
\begin{environment-name}
< environment body (usually text or mathematics) >
\end{environment-name}
```

Centering Text, the center Environment:

```
\begin{center}
This paragraph is centered.
\end{center}
```

produces

This paragraph is centered.

Lists, the enumerate Environment:

```
\begin{enumerate}
\item This is the first item.
\item This is the second item.
\item This is the third item.
\end{enumerate}
```

produces

1. This is the first item.
2. This is the second item.
3. This is the third item.

```
\begin{enumerate}
\item This is the first item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\end{enumerate}
\item This is the second item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\item This is the third subitem.
\end{enumerate}
\item This is the third item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\end{enumerate}
\end{enumerate}
```

produces

1. This is the first item.
    (a) This is the first subitem.
    (b) This is the second subitem.
2. This is the second item.
    (a) This is the first subitem.
    (b) This is the second subitem.
    (c) This is the third subitem.
3. This is the third item.
    (a) This is the first subitem.
    (b) This is the second subitem.

and

```
\begin{enumerate}
\item This is the first item.
\begin{enumerate}
\item This is the first subitem.
\begin{enumerate}
\item This is the first subsubitem.
\item This is the second subsubitem.
```

```
\end{enumerate}
\item This is the second subitem.
\end{enumerate}
\item This is the second item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\begin{enumerate}
\item This is the first subsubitem.
\item This is the second subsubitem.
\begin{enumerate}
\item This is the first subsubsubitem.
\item This is the second subsubsubitem.
\end{enumerate}
\end{enumerate}
\item This is the third subitem.
\end{enumerate}
\item This is the third item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\end{enumerate}
\end{enumerate}
```

produces

1. This is the first item.
    (a) This is the first subitem.
        i. This is the first subsubitem.
        ii. This is the second subsubitem.
    (b) This is the second subitem.
2. This is the second item.
    (a) This is the first subitem.
    (b) This is the second subitem.
        i. This is the first subsubitem.
        ii. This is the second subsubitem.
            A. This is the first subsubsubitem.
            B. This is the second subsubsubitem.
    (c) This is the third subitem.
3. This is the third item.

(a) This is the first subitem.

(b) This is the second subitem.

but I digress.

More Lists, the itemize Environment:

```
\begin{itemize}
\item This is the first item.
\item This is the second item.
\item This is the third item.
\end{itemize}
```

produces

- This is the first item.
- This is the second item.
- This is the third item.

```
\begin{itemize}
\item This is the first item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\end{itemize}
\item This is the second item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\item This is the third subitem.
\end{itemize}
\item This is the third item.
\begin{enumerate}
\item This is the first subitem.
\item This is the second subitem.
\end{itemize}
\end{itemize}
```

produces

- This is the first item.
  - This is the first subitem.
  - This is the second subitem.

- This is the second item.
  - This is the first subitem.
  - This is the second subitem.
  - This is the third subitem.
- This is the third item.
  - This is the first subitem.
  - This is the second subitem.

and

```
\begin{itemize}
\item This is the first item.
\begin{itemize}
\item This is the first subitem.
\begin{itemize}
\item This is the first subsubitem.
\item This is the second subsubitem.
\end{itemize}
\item This is the second subitem.
\end{itemize}
\item This is the second item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\begin{itemize}
\item This is the first subsubitem.
\item This is the second subsubitem.
\begin{itemize}
\item This is the first subsubsubitem.
\item This is the second subsubsubitem.
\end{itemize}
\end{itemize}
\item This is the third subitem.
\end{itemize}
\item This is the third item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\end{itemize}
\end{itemize}
```

produces

- This is the first item.
  - This is the first subitem.
    - This is the first subsubitem.
    - This is the second subsubitem.
  - This isthe second subitem.
- This is the second item.
  - This is the first subitem.
  - This is the second subitem.
    - This is the first subsubitem.
    - This is the second subsubitem.
      - This is the first subsubsubitem.
      - This is the second subsubsubitem.
  - This is the third subitem.
- This is the third item.
  - This is the first subitem.
  - This is the second subitem.

but I digress, again.

Of course you can combine these:

```
\begin{itemize}
\item This is the first item.
\begin{enumerate}
\item This is the first subitem.
\begin{itemize}
\item This is the first subsubitem.
\item This is the second subsubitem.
\end{itemize}
\item This is the second subitem.
\end{enumerate}
\item This is the second item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\begin{enumerate}
\item This is the first subsubitem.
\item This is the second subsubitem.
\begin{enumerate}
```

```
\item This is the first subsubsubitem.
\item This is the second subsubsubitem.
\end{enumerate}
\end{enumerate}
\item This is the third subitem.
\end{itemize}
\item This is the third item.
\begin{itemize}
\item This is the first subitem.
\item This is the second subitem.
\end{itemize}
\end{itemize}
```

produces

- This is the first item.
    1. This is the first subitem.
        - This is the first subsubitem.
        - This is the second subsubitem.
    2. This is the second subitem.
- This is the second item.
    - This is the first subitem.
    - This is the second subitem.
        1. This is the first subsubitem.
        2. This is the second subsubitem.
            (a) This is the first subsubsubitem.
            (b) This is the second subsubsubitem.
    - This is the third subitem.
- This is the third item.
    - This is the first subitem.
    - This is the second subitem.

Even More Lists, the description Environment:

```
\begin{description}
\item [Integer] --- any number from the set $\{ 0, \pm 1, \pm 2, \ldots \}$.
\item [Rational] --- any number of the form $\frac{a}{b}$ where $a$ and $b$
are integers with $b \neq 0$.
\item [Group] --- Let $G$ be a nonempty set together with a binary
opertion (usually called multiplication) that assigns to each pair
```

```
$(a,b)$ of elements of $G$ an element of $G$ denoted $ab$.  We say
$G$ is a {\em group} under this operation if the following three
properties are satisfied.
\begin{enumerate}
\item {\em Associativity.} The operation is associative; that is,
$(ab)c = a(bc)$ for all $a$, $b$, $c$ in $G$.
\item {\em Identity.} There is an element $e$ (called the {\em identity})
in $G$ such that $ae = ea = a$ for all $a$ in $G$.
\item {\em Inverses.} For each element $a$ in $G$, there is an element
$b$ in $G$ (called an {\em inverse} of $a$) such that $ab = ba = e$.
\end{enumerate}
\end{description}
```

produces

Integer — any number from the set $\{0, \pm 1, \pm 2, \dots\}$.

Rational — any number of the form $\frac{a}{b}$ where a and b are integers with $b \neq 0$.

Group — Let G be a nonempty set together with a binary operation (usually called multiplication) that assigns to each pair (a, b) of elements of G an element of G denoted ab. We say G is a *group* under this operation if the following three properties are satisfied.

1. *Associativity.* The operation is associative; that is, $(ab)c = a(bc)$ for all a, b, c in G.
2. *Identity.* There is an element e (called the *identity*) in G such that $ae = ea = a$ for all a in G.
3. *Inverses.* For each element a in G, there is an element b in G (called an *inverse* of a) such that $ab = ba = e$.

Figures, the figure Environment:

```
\begin{figure}[htbp]
The body of the figure would go here.
\caption{Not really a figure, more like text.}
\end{figure}
```

produces

The body of the figure would go here.

Figure 1: Not really a figure, more like text.

The figure environment, as with a few others, are floating environments. This means that if there is not enough room for the figure's contents, LaTeX will automatically move it to another position. The optional placement argument (in []) determines where LaTeX will try to place your figure. There are four places where LaTeX can possibly put a float:

The body of the second figure. This figure was placed before the sentence. "The optional placement argument (in []) determines where LaTeX will try to place your figure." above.

Figure 2: Look where I am.

1. h — (Here) at the position in the text where the figure environment appears.
2. t — (Top) at the top of a text page.
3. b — (Bottom) at the bottom of a text page.
4. p — (Page of floats) on a separate float page, which is a page containing no text, only floats.

LaTeX will try to place the figure in your order of preference. That is, the list `[htbp]` means place the figure here if there is enough room, like it did with Figure 1. If there is not enough room, place it at the top of a text page (usually the next page), as it did with Figure2. If there is not enough room for that, place it at the bottom of a text page (usually the next page). Finally, if there is not enough room for any of those place the figure on a page that contains only floating structures (like figures).

Tables, the tabular Environment:

There are several environments that can be used for making tables, one of the easier ones to set up is the tabular environment. With this environment we select a layout by columns. We use an l for a column of left-justify the text, r for right-justified text, c for centered text and — for a vertical line. There is another tag, pwidth, for a column containing justified text with line breaks. To place a horizontal line in the table we use the command `\hline`. When you are creating your table, place `&` between the entries of the table (just like arrays and equation arrays). Also place `\\` at the end of each line. For example,

```
\begin{tabular}{|r|l|}
\hline
7C0 & hexadecimal \\
3700 & octal \\
11111000000 & binary \\
\hline \hline
1984 & decimal \\
\hline
\end{tabular}
```

produces

7C0	hexadecimal
3700	octal
11111000000	binary
1984	decimal

There are a couple other tags that can be used in the tabular environment. The two we will look at are `@{·}` and `\multicolumn`. The `@{·}` is used in the layout portion of the table and it is used to put a specific character or symbol as the divider between the two columns. It would be used in place of the vertical line, but unlike the vertical line it kills the inner-column spacing. For example,

```
\begin{tabular}{c | c @{$\vdots$} c}
\hline
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\hline
\end{tabular}
```

produces

1	2⋮3
4	5⋮6
7	8⋮9

The `\multicolumn` command is used to merge columns together. For example,

```
\begin{tabular}{c r @{.} l}
Pi expression & \multicolumn{2}{c}{Value} \\
\hline
$\pi$ & 3&1416 \\
$\pi^{\pi}$ & 36&46 \\
$(\pi^{\pi})^{\pi}$ & 80662&7 \\
\end{tabular}
```

produces

Pi expression	Value
π	3.1416
π^{π}	36.46
$(\pi^{\pi})^{\pi}$	80662.7

Notice how the heading Value is centered over two columns of the table.

A Few Miscellanious LaTeX Things:

Dashes:

There are four types of dashes in LaTeX. The hyphen, numeric range dash, em-dash and the negative sign. The hyphen is done with a single `-`, the numeric range dash is done with two `--`, em-dash with three `---`, and the negative sign is done with one and is printed only in math mode. For example,

LaTeX	Printed Text
`multi-line`	multi-line
`Pages 10--23`	Pages 10–23
`This is it---so it is.`	This is it—so it is.
`$-7$`	-7

Quotation Marks:

Quotiation marks are a little different in LaTeX you must specify which way they go. For a single quote you must use the graves quote to begin the quote and a regular single quote to end. The graves quote is below the tilde in the upper left of the keyboard. For example, `‘This is a quote.’` produces ‘This is a quote.’ Double quotations are done using two single quotes. Again the graves quotes start and the regular ones finish. For example, `‘‘This is a quote.’’` produces “This is a quote.”

Dots, Dots and More Dots:

In math mode we have vertical dots, horizontal dots (both low and middle) and diagonal dots. The lower horizontal dots can also be used outside of math mode. For example,

```
\[
x_1 + x_2 + x_3 + \cdots + x_n = y
\]
```

produces

$$x_1 + x_2 + x_3 + \cdots + x_n = y$$

```
\[
\{ x_1, x_2, x_3, \ldots \}
\]
```

produces

$$\{x_1, x_2, x_3, \ldots\}$$

```
\[
\begin{array}{cccc}
a_{11} & a_{12} & \cdots & a_{1n} \\
a_{21} & a_{22} & \cdots & a_{2n} \\
\vdots & \vdots & \ddots & \vdots \\
a_{m1} & a_{m2} & \cdots & a_{mn} \\
\end{array}
\]
```

produces

$$\begin{matrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{matrix}$$

and

```
and so on \ldots
```

produces

and so on ...

Footnotes:

Footnotes are easy[1]. The last footnote was produced by

```
Footnotes are easy\footnote{They are practically automatic!}
```

Note that the footnote number is produced automatically as well as the line and footnote at the bottom of the page.

[1] They are practically automatic!

TEXnique #5

Graphics:

Including graphics into LaTeX is not as easy as including graphics into other word processing systems, like Word. Although it requires a little doing, the graphics that are produced are of the highest quality. One thing you will notice about LaTeX graphics is that the image is well preserved when it is scaled up or down. This is not always the case with Word.

There are several types of graphics files than can be loaded into a LaTeX document. The most common formats used today are the bitmap and jpeg. The type of graphics you load into the document will depend on what you intend to do with the document. If you wish to simply print the document out to a standard printer then you want to use bitmaps. If, on the other hand, you want to create a pdf file of the document and display the document on the web then you must use the jpeg format. We will cover both methods here. In either case, the graphic files must be in the same directory as the LaTeX document.

Including Bitmap Graphics for Printing:

To include a bitmap image do the following.

1. Include the following statement in the preamble of your document.

   ```
   \usepackage{graphicx}
   ```

2. At the position in the document you wish to include the graphic insert the command

   ```
   \includegraphics[height=<height>pt,width=<width>pt]{<filename>.bmp}
   ```

   where `<filename>` is the filename of the graphic file, `<width>` is the width of the image and `<height>` is the height of the image. I usually set the width and height to $\frac{1}{2}$ the number of points that the image is in pixels. For example, to include the image file `ex3p1.bmp` with a height of 143 points and a width of 193 points we would use the command.

   ```
   \includegraphics[height=143pt,width=193pt]{ex3p1.bmp}
   ```

   The image's size, in pixels, was 286 by 386. You can use whatever height and width you wish but you should keep the ratios the same. If you do not, the image will appear to be stretched.

   I would also suggest that you place the image in an environment such as a figure or displayed mathematics. For example,

   ```
   \begin{figure}[hbt]
   \centering
   % height=203pt,width=286pt
   \includegraphics[height=101pt,width=143pt]{e1p1.bmp}
   \caption{$\mathbf{r}(t) = \langle t\cos(t), t\sin(t), 0 \rangle$}
   \end{figure}
   ```

   or

   ```
   \[
   % height=203pt,width=286pt
   \includegraphics[height=101pt,width=143pt]{e1p1.bmp}
   \]
   ```

3. Compose the document.
4. Use YAP to view and print the document.

Including JPEG Graphics for PDF Files:

I tend to use this method most often since most of the documents I write make their way to the web and since Adobe Reader will print them out for me anyway. The only inconvenience is that the YAP previewer will not display the images but it will at least leave the same amount of space for them.

To include jpeg graphics into your document do the following.

1. Include the following statement in the preamble of your document.

   ```
   \usepackage[dvipdfm]{graphicx}
   ```

2. Create a bounding box for the graphic. To do this simply click on the Establish Bounding Box icon on the toolbar (two crossing yellow rulers) or select Edit ≻ Establish Bounding Box for JPG. A dialog box will appear giving you a listing of all the jpg files in the current directory. Simply double-click the filename of the file you want to include. When you do, the dialog box will disappear and a DOS box (black background) will appear for a split second. When it disappears the bounding box has been created. If you look at the contents of the directory you will see that another file has been created with the same name as the graphic but with a BB extension. You never need to edit this file, it is simply for the program to use.

3. At the position in the document you wish to include the graphic insert the command

   ```
   \includegraphics[height=<height>pt,width=<width>pt]{<filename>.jpg}
   ```

   where `<filename>` is the filename of the graphic file, `<width>` is the width of the image and `<height>` is the height of the image. I usually set the width and height to $\frac{1}{2}$ the number of points that the image is in pixels. For example, to include the image file `ex3p1.jpg` with a height of 143 points and a width of 193 points we would use the command.

   ```
   \includegraphics[height=143pt,width=193pt]{ex3p1.jpg}
   ```

   The image's size, in pixels, was 286 by 386. You can use whatever height and width you wish but you should keep the ratios the same. If you do not, the image will appear to be stretched.

   I would also suggest that you place the image in an environment such as a figure or displayed mathematics. For example,

   ```
   \begin{figure}[hbt]
   \centering
   % height=203pt,width=286pt
   \includegraphics[height=101pt,width=143pt]{e1p1.jpg}
   \caption{$\mathbf{r}(t) = \langle t\cos(t), t\sin(t), 0 \rangle$}
   \end{figure}
   ```

   or

   ```
   \[
   % height=203pt,width=286pt
   \includegraphics[height=101pt,width=143pt]{e1p1.jpg}
   \]
   ```

4. Compose the document.
5. PDF the document by clicking on the Create PDF File icon on the toolbar or by selecting LaTex ≻ Create PDF File from the menu.
6. Use Adobe Acrobat Reader to view and print the document.

TEXnique #6

Do It Yourself:

One interesting feature that LaTeX has is that you can define your own commands. So if you want to do something special you simply make a command to do it. The TeX system also has ways for you to create your own fonts but we will not need to go quite that far. Creating commands, on the other hand can come in handy quite often. The general syntax for a new command is

```
\newcommand{name}[num]{definition}
```

Where the name is the command name. Command names must start with a backslash \ and can not begin with `\end`. Furthermore, the command can not already exist. The num is the number of arguments the command can take and the definition is the way that the new command will act. For example, if we were to define,

```
\newcommand{\x}{$x$}
```

Then `\x` would produce x. LaTeX simply takes `$x$` and uses it in place of the `\x`. This was an example of a command without any arguments. If we had used

```
\newcommand{\x}[1]{$x^{#1}$}
```

Then `\x{3}` would produce x^3. We can use as many arguments as we would like, simply place each in curly brackets. For example, with the new command,

```
\newcommand{\x}[2]{$x_{#1}^{#2}$}
```

`\x{1}{3}` would produce x_1^3. New commands need not be mathematical in nature. For example, the command

```
\newcommand{\names}[2]{#1 #2's name written formally would be #2, #1}
```

would make `\names{Don}{Spickler} and \names{John}{Doe}.` produce, Don Spickler's name written formally would be Spickler, Don and John Doe's name written formally would be Doe, John. One command I create frequently is for binomial coefficients. The command,

```
\newcommand{\binom}[2]{\left( \begin{array}{cc} #1 \\ #2 \\ \end{array}\right)}
```

will make `$\binom{n}{t}$` produce $\binom{n}{t}$ and

```
\[
\sum_{i = 0}^n \binom{n}{i} = 2^n
\]
```

produce

$$\sum_{i=0}^{n} \binom{n}{i} = 2^n$$

You can also redefine a command that already exists by using the redefine command. The general syntax for the redefine command is

```
\renewcommand{name}[num]{definition}
```

where the name, num and definition are the same as with newcommand. For example,

```
\newcommand{\x}{$x$}
Consider the variable \x.
\renewcommand{\x}{$X$}
Consider the variable\x.
```

produces, Consider the variable x. Consider the variable X.

Be careful not to redefine a LaTeX command unless you really want to. Most of the time you will place all of your new commands in the preamble to the document but technically they can go anywhere in the document.

You can also create your own environments with the `\newenvironment` and the `\renewenvironment` commands. The general syntax for these commands is,

```
\newenvironment{name}[num]{begdef}{enddef}
\renewenvironment{name}[num]{begdef}{enddef}
```

The name is the name of the environment, the num is the number of arguments, begdef is the stuff to do before the text that will be included in the environment and enddef is the stuff to do after the text that will be included in the environment. For example, the environment

```
\newenvironment{cards}
{\vskip 10pt \noindent $\diamondsuit\clubsuit\heartsuit\spadesuit$ \hfill}
{\hfill $\spadesuit\heartsuit\clubsuit\diamondsuit$ \vskip 10pt}
```

with

```
\begin{cards}
Pick a card, any card.
\end{cards}
```

will produce,

♢♣♡♠ Pick a card, any card. ♠♡♣♢

The next several pages explain some of the more advanced features in LaTeX when you are working with larger projects, such as reports or books that have multiple chapters or sections. When you are doing a bigger project you will need to use a different document class. So far all we have used is the article. That is, our first line to the LaTeX document was something like,

```
\documentclass[10pt]{article}
```

For larger projects we need to use `report` or `book` instead of `article`. This will allow you to use the chapter, section, subsection and subsubsection features as well as the automatic title page, contents list and figure list.

You can also set up your own counters that can work either independently of the built-in counters or integrated with the built-in counters.

The following LaTeX code created the table of contents and the first two sections of chapter 1. We will examine each of the features one at a time.

```
\tableofcontents

\chapter{More Advanced Features}

\section{You can count on \LaTeX{}}

\vskip 10pt

Actually we should say that \LaTeX{} can count for you.  As you
```

```
saw in a previous \TeX{}nique, \LaTeX{} can automatically
number lists, footnotes, equations, figures and pages.  There are
many more
types of automatic numbering built into \LaTeX{}.  Some of these
include, chapter, section and subsection numbering as well as
table, figure and paragraph numbering.

\newtheorem{lemma}{Lemma}[section]

\begin{lemma}
There is always more than one way to prove a mathematical theorem.
\label{lemma:first}
\end{lemma}

\begin{lemma}
There are two or more ways to prove any  mathematical theorem.
See lemma \ref{lemma:first} on page \pageref{lemma:first}.
\end{lemma}

\newtheorem{thm}{Theorem}[chapter]

\begin{thm}
What can go wrong will go wrong.
\label{thm:Murphy}
\end{thm}

\newtheorem{cor}{Corollary}[page]

\begin{cor}
What possibly can go wrong will possibly go wrong.  See Theorem
\ref{thm:Murphy}
\end{cor}

\newcounter{don_s_counter_1}
\stepcounter{don_s_counter_1}
\newcounter{don_s_counter_2}[section]
\stepcounter{don_s_counter_2}

\newenvironment{dlaw}
{\noindent{\bf Don's Law \arabic{don_s_counter_1}:} }
{\vskip 10pt \stepcounter{don_s_counter_1}}

\newenvironment{drule}
{\noindent{\bf Don's Rule \arabic{don_s_counter_2}:} }
{\vskip 10pt \stepcounter{don_s_counter_2}}

\begin{dlaw}
It always takes twice the amount of time to complete a project as
I think it will take.
\end{dlaw}

\begin{drule}
The weekend is a time to get caught up on work.  So why am I more
behind on Monday than I was the previous Friday?
```

```
\end{drule}

\section{Next Section}

\begin{dlaw}
People seldom remember how quickly you finish a job but they always
remember how well you did the job.
\end{dlaw}

\begin{drule}
The weekend is a time to relax and be rejuvenated for the next
week.  So why am I more tired on Monday than I was the previous Friday?
\end{drule}
```

Contents

Chapter 1

More Advanced Features

1.1 You can count on LaTeX

Actually we should say that LaTeX can count for you. As you saw in a previous TeXnique, LaTeX can automatically number lists, footnotes, equations, figures and pages. There are many more types of automatic numbering built into LaTeX. Some of these include, chapter, section and subsection numbering aswell as table, figure and paragraph numbering.

Lemma 1.1.1 *There is always more than one way to prove a mathematical theorem.*

Lemma 1.1.2 *There are two or more ways to prove any mathematical theorem. See lemma 1.1.1 on page 6.*

Theorem 1.1 *What can go wrong will go wrong.*

Corollary 6.1 *What possibly can go wrong will possibly go wrong. See Theorem 1.1*

Don's Law 1: It always takes twice the amount of time to complete a project as I think it will take.

Don's Rule 1: The weekend is a time to get caught up on work. So why am I more behind on Monday than I was the previous Friday?

1.2 Next Section

Don's Law 2: People seldom remember how quickly you finish a job but they always remember how well you did the job.

Don's Rule 0: The weekend is a time to relax and be rejuvenated for the next week. So why am I more tired on Monday than I was the previous Friday?

1.3 Let's Look at What We Did to Create the Last Two Sections

First the command `\tableofcontents` takes the chapter and section commands throughout the document and automatically produces a table of contents from them. The Contents title is produced automatically as is the headings and page number layout.

The command `\chapter{More Advanced Features}` produces a new chapter by skipping to the next page, placing the Chapter heading, number and title at the top of the page.

The command `\section{You can count on \LaTeX{}}` produces a new section. Notice that the first paragraph of the section is not indented. This is also done automatically since it is the current convention in printed materials.

The next command of interest is the `\newtheorem` command. This command creates another environment similar to the `theorem` environment. That is, you can produce automatically numbered theorems, lemmas, corollaries, ... simply with a single command and a begin/end environment structure. For example, the command

```
\newtheorem{lemma}{Lemma}[section]
```

creates an environment `lemma` that produces a bold first word of **Lemma** and is labeled with the Chapter/Section numbering before the lemma number. In general the syntax is

```
\newtheorem{env_name}{caption}[within]
```

where `env_name` is the name of the environment, `caption` is the caption the is before the numbering and `within` is the numbering scheme that is before the number of the item. You can use any of the counters available in LaTeX for the `within`, listed below, as well as any counter you create.

```
part            paragraph       figure          enumi
chapter         subparagraph    table           enumii
section         page            footnote        enumiii
subsection      equation        mpfootnote      enumiv
subsubsection
```

It is most common to use one of the following.

```
part
chapter
section
subsection
subsubsection
```

So with the lemma environment in our example, the environment,

```
\begin{lemma}
There is always more than one way to prove a mathematical theorem.
\end{lemma}
```

produces

Lemma 1.3.1 *There is always more than one way to prove a mathematical theorem.*

The 1.3.1 tells us that it is in Chapter 1, section 3 and is the first lemma in that section.

Notice that the other theorem environments are numbered by the chapter and page.

```
\newtheorem{thm}{Theorem}[chapter]
\newtheorem{cor}{Corollary}[page]
```

Notice the differences between the output of these environments and that of the lemma.

The next new topic that is in this document is cross-referencing. Notice the three commands

`\label{lemma:first}`, `\ref{lemma:first}` and `\pageref{lemma:first}`

The label command has the general syntax `\label{labelref}` where `labelref` is any label you want. You could use lemma1, FundThmofFinAbelGroups, GreensThm or even George. I would recommend that you use something that is not easily duplicated and is representative of the theorem you are referencing. The general syntax of the ref command is `\ref{labelref}`. This command will print the current value of the number of the theorem that is referenced by `labelref`. Finally, the syntax for the pageref command is `\pageref{labelref}`. This command will print the current page number that the theorem that is referenced by `labelref` is on. You may think that this is a bit of overkill but this little extra work can save hours when you are editing a large work. For example, say you did not use these cross referencing features but instead typed in the theorem numbers and page numbers by hand. What would happen if you decided to move a

chapter, interchange a couple sections, combine two chapters into one or add a chapter or two in the middle of the document? What would happen is that you would have to read through each page of the document and edit all of the references. If you used the cross referencing features all you would need to do is recompose the document and all of the changes would be made automatically. If you are headed to graduate school you will be writing documents of this magnitude and be doing major editing to them.

Creating new counters is easy and useful, especially if you do not want to use the default numbering systems available with LaTeX. To create a new counter simply use the command `\newcounter{counter}` or `\newcounter{counter}[numscheme]` where `counter` is the name of the new counter and `numscheme` is the numbering style that resets your counter when it changes. For example,

```
\newcounter{don_s_counter_1}
```

creates a new counter named `don_s_counter_1`. Since there is no `numscheme` the counter keeps incrementing throughout the document and does not reset, unless you reset it with the setcounter command. The command

```
\newcounter{don_s_counter_2}[section]
```

creates a new counter named `don_s_counter_2` and resets each time the section number is updated, that is, each time we move to a new section of the document.

You need to update your own counters manually. To update a counter use the stepcounter command. For example,

```
\stepcounter{don_s_counter_1}
```

increments the `don_s_counter_1` counter. To print the value of the counter at any time in the document use either the `\alph`, `\arabic`, `\roman` or `\fnsymbol` commands. For example, the code

```
\newenvironment{dlaw}
{\noindent{\bf Don's Law \arabic{don_s_counter_1}:} }
{\vskip 10pt \stepcounter{don_s_counter_1}}
```

created a new environment called `dlaw`. In the pretext portion of the definition the command

```
\arabic{don_s_counter_1}
```

prints the counter number using Aribic numbers. In the posttext portion of the definition the command `\stepcounter{don_s_counter_1}` increments the counter. Note that counters all start at and are reset to 0.

Whenever you use any counting environment, like theorems, chapters or our own counters you must compose the document several times, either two or three. This is because when you have a counter or cross reference the first composition of the file will write information to the disk containing the reference numbers and page numbers that are needed. The second composition of the document will use the saved data to fill in the appropriate references in the document. There are some references that require a third composition but most of the time two will be sufficient.

1.4 Including External LaTeX Files into a Document

When your project gets really big, like a book, you will want to break the document up into smaller files and paste them together at the end. For example when you are writing a book you may want to place each chapter or section in its own file and then have the computer join them together into a book when you are finished. LaTeX offers a very simple way to do this using the `\include` and the `\input` commands.

The `\input{file}` command causes the indicated `file` to be read and processed, exactly as if its contents had been inserted in the current file at that point. The file name may be a complete file name with extension or just a first name, in which case the file extension of `.tex` is used.

The `\include{file}` command is similar to the `\input{file}` command except that included files are automatically placed on the next page of the document. That is, the `\include{file}` command is equivalent to

```
\clearpage \input{file} \clearpage
```

The `\include{file}` command can also be used in conjunction with the `\includeonly` command for selective inclusion of files. If file is one the file names in the file list of the `\includeonly` command the `\include` command is again equivalent to

```
\clearpage \input{file} \clearpage
```

except that if the file included file does not exist, then a warning message rather than an error is produced. If the file is not in the file list, the `\include` command is equivalent to `\clearpage`. Unlike the `\input` command, the `\include` command may not appear in the preamble or in a file read by another `\include` command.

The `\includeonly{file_list}` command controls which files will be read in by an `\include` command. The `file_list` should be a comma-separated list of filenames. Each filename must match exactly a filename specified in a `\include` command. This command can only appear in the preamble of the document.

For example,

```
\input{section1_1.tex}
```

will read the text from the `section1_1.tex` file and place it at the location of the input command. The command

```
\include{section1_1}
```

will do the same thing except that the contents of the `section1_1.tex` file will begin on a new page. The code

```
\documentclass[10pt]{report}
\includeonly{section1,section3}
\begin{document}

\include{section1}
\include{section2}
\include{section3}

\end{document}
```

will include only the files section1.tex and section3.tex.

Including Images In Your LaTeX Documents
Using The TeXnicCenter Editor[1]

Including graphics into LaTeX is not as easy as including graphics into other word processing systems, like Word. Although it requires a little doing, the graphics that are produced are of the highest quality. One thing you will notice about LaTeX graphics is that the image is well preserved when it is scaled up or down. This is not always the case with Word. The most common way to include a graphics file is to save the graphics file in one of the supported types (PNG, BMP, JPG, GIF or PDF) and use the `LaTeX => PDF` output profile. This will produce a PDF file with the graphic included and then using Adobe Acrobat Reader or other PDF reader you can view and print the document.

Say we have the following image, `Pic01.jpg`, that we want in our document.

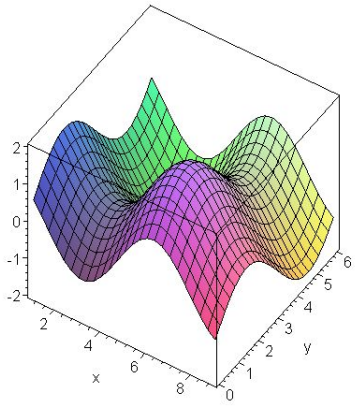

Figure 1: Pic01.jpg

1. Set the output profile to `LaTeX => PDF`.
2. Include the following command in the preamble of your document.

   ```
   \usepackage[pdftex]{graphicx}
   ```

3. At the position in the document where you wish to include the graphic insert the command

   ```
   \includegraphics[scale=<factor>]{<filename>}
   ```

   where `<filename>` is the filename of the graphic file and `<factor>` is the scaling factor of the image. For example, to include the image file `Pic01.jpg` with a scaling factor of 0.35 we would use the command.

   ```
   \includegraphics[scale=0.35]{Pic01.jpg}
   ```

   In place of the scale option you can use any of the options available for the includegraphics command. You should place the image in an environment such as a figure, centered or displayed mathematics. For example,

[1] This document assumes that you have installed the LaTeX typesetting distribution from the Salisbury University Math and Computer Science Department which was bundled in 2008. In particular, you are using pdflatex as the compiler for the PDF output profile.

```
\begin{figure}[hbt]
\centering
\includegraphics[scale=0.35]{Pic01.jpg}
\caption{Pic01.jpg}
\end{figure}
```

or

```
\begin{center}
\includegraphics[scale=0.35]{Pic01.jpg}
\end{center}
```

or

```
\[
\includegraphics[scale=0.35]{Pic01.jpg}
\]
```

4. Compose and view the document.