LATEX (second session)

REMINDER OF THE BASIC PROCEDURES:

Go to your BLACKBOARD, to "Professional Skills...", open "LATEX" folder in the "Learning materials"; download all the files to a folder LATEX (create a new folder) on this computer. Or just one 'all files' zip file.

Double-clicking on a file with extension '.tex' will open it with TEXNICCENTER. (If not, open the editor TEXNICCENTER on the desktop (or locate it in the 'All programs' or 'All programs\Latex).

Make sure 'Output profile' window shows 'LATEX ⇒ DVI — can be chosen in the drop-down menu. (There are also unimportant options how the application looks: in menu 'View' in 'Application Look' choose Windows-7, or Office2007, say.) There may be a left panel "Outline", "Environments", "Files", which can be closed.

Pressing 'Build current file' button (with blue down-arrow, second on the right of 'Output profile' window). If it works OK, then we can view the resulting DVI by pressing 'View output' button (second on the right of 'Build', with 'magnifying glass'). **All the work is done in the editor;** if any changes are needed, we must go back to the input file in the editor (=TEXNICCENTER), correct, then build DVI again to see what we get. When we 'Build', the bottom panel informs about compilation (=building=typesetting) process, errors. Should be 0 errors. View the result by pressing 'View' ('magnifier' icon). Bottom panel can be made smaller.

On a wide-screen monitor it is convenient to arrange the window with TEXNICCENTER as one half, and the DVI-viewer as another.

If there are errors, usually does not stop and some output is produced (if not too severe errors!), but red icon appears, click 'next error' (left of red buttons) will give you red indicator on the left in your file, so you can make correction at this point. Best correct the first error then build again, as often subsequent errors are caused by previous ones. If still there are errors, again click on 'Next error', etc.

Typical error: braces in commands (specifying the scope) are missing....

Numbering and labelling. Many things are automatically numbered (like sections), all numbered elements may be labelled:

```
\section{\label{s1} Intro}
```

Then can be used by `\ref{s1}` command, like: `In Section \ref{s1} we...`

1 Mathematical formulae.

Open 3third.tex, read about mathematical formulae. Build DVI, view the result.

Mathematical formulae in LATEX are between $ signs. (There should be no extra $ signs within a formula!) Some manuals also say between `\(` and `\)`, which produces the same result, this is actually better as parenthesis match, but $ is simpler. In accordance to maths writing standards, letters become italicized, operations +, =, etc. have proper spaces around them, Greek letters can be used, many special symbols, etc. Simplest: $x + y = z - \alpha$.

Symbols that can be used in maths formulae directly from the keyboard:

+ − = ! / ( ) [ ] < > | ′ :

Non-strict inequalities: `\leq` and `\geq` produce $\leq$ and $\geq$. E.g.: $2x + 1 \leq 3$.

Most common constructs:

Power exponents (=superscripts): by `^{}`, for example, x^2 is produced by `$x^{2}$`;

more: $(x - y)^{a^2-bz}$ from `$(x-y)^{a^{2}-bz}$`.

Indices (=subscripts): by `_{}`, like x_i or $a_{i,j}$, or b_{i_1} are produced by `$x_{i}$`, `$a_{i,j}$`, `$b_{i_{1}}$`, resp.

Subscripts and superscripts can be combined: $a_{ij}^2 = a_{ij}^2$ from `$a_{ij}^{2}=a^{2}_{ij}$`.

Fractions: `\frac{}{}`, like $\frac{1}{2}$ or $\frac{x-2}{(x^2-b)(a+b)}$ are produced by `$\frac{1}{2}$`, `$\frac{x-2}{(x^{2}-b)(a+b)}$`

Roots: `\sqrt{...}` for square root of ..., or more general `$\sqrt[n]{...}$` for the n-th root. E.g. $\sqrt{x - a}$ from `$\sqrt{x-a}$`, or $\sqrt[3]{-1}$ from `$\sqrt[3]{-1}$`,
or even $\sqrt[4]{\frac{a+b}{c+\sqrt{d}}}$ from `$\sqrt[4]{\frac{a+b}{c+\sqrt{d}}}$`.
. . . . . .
. . . . . .

2 Environments.

One of the important features of LATEX. Begin with `\begin{...}` and end with `\end{...}`. There are standard environments, like theorem-type or equation. Usually generates a number (or no number with $*$). For cross-references include `\label{..}`.

Equation: input

```
\begin{equation} \label{lbl}
a=b
\end{equation}
```

produces

$$a = b. \tag{1}$$

Then we can refer to it by "In equation (1)..." by using `(\ref{lbl})`. But much smarter using `\eqref{lbl}`, which only works if `\usepackage{amsmath}` is used in top-matter — `\eqref` is better since it makes (..) and makes it always upright.

The `\usepackage{amsthm}` also provides nice environment `align`, which makes nice breaks of formulae (read manuals if interested).

Theorem-type: These environments must be declared in 'top-matter'. Top-matter is stuff after `\documentclass...` and `\begin{document}`.

```
\newtheorem{thrm}{Theorem},
```

where first parameter in braces `{thrm}` is the name (you choose) of this theorem-type environment that you use in the input file, and second parameter `{Theorem}` tells LATEX what you choose to print as a title, followed by number. For example, one usually also declares

```
\newtheorem{lemma}{Lemma}
\newtheorem{crly}{Corollary}
```

This theorem environment makes vertical space around statement, italicizes the statement, gives title like Theorem 1, Lemma 2, etc.

Example (only works if we declared in top-matter `\newtheorem{thrm}{Theorem}`!!!):

```
\begin{thrm}\label{th1}
If $f$ is a function whose derivative exists at every point, then $f$
is a continuous function.
\end{thrm}
```

produces

Theorem 1. *If f is a function whose derivative exists at every point, then f is a continuous function.*

Another example: with declaration in top-matter `\newtheorem{lemma}{Lemma}`, the input

```
\begin{lemma}\label{l1}
If $f$ is a function whose derivative exists at a point $x_0$, then $f$
is continuous at this point.
\end{lemma}
```

gives output

Lemma 1. *If f is a function whose derivative exists at a point x_0, then f is continuous at this point.*

Example how to refer: "We saw in Lemma 1 that..." (using `Lemma \ref{l1}`).

Proof: if using `\usepackage{amsthm}`, there is a special environment 'proof':
`\begin{proof}`
.....
`\end{proof}`
produces nice spacing and q.e.d. (quod erat demostrandum = what was required to prove) symbol at the end:

Proof. Since ... we have ...
Then..... whatever. The result follows. □

EXERCISE: Open 4fourth.tex, read, build (run build **twice!**), view. Save as my4fourth.tex. Open 4fourth-r.pdf. Writing over my4fourth.tex, produce input file such that output would match 4fourth-r.pdf. **Use environments, labels, and `\ref{..}` and `\eqref{..}` commands, not 'manual' numbers!** (Remember to run build **twice** to get cross-references right.)

3 Tables.

Environment `\begin{tabular}{<parameters>}.. \end{tabular}` produces simple tables. Here 'parameters' is a sequence of symbols l, r, c in any order (meaning left-, right-justified, centred columns). Optionally vertical line `|` between these l, r, c makes vertical divider. In the input file, in a given row, column elements are separated by the special symbol &, and a new row is started by \\. Horizontal lines are produced by `\hline` (between lines). For example, if we use `\begin{tabular}{|r|cl|}.. \end{tabular}`, then we get three columns, vertical line between first and second columns, and text is right-justified in 1st column, centred in 2nd, left-justified in third:

Maths	23	students
Physics	12	PhDs
Comp. Science	123 staff	mixed
Others		

Tabular environment can be used without lines to simply arrange text in justified columns, like

BSc Maths	Bsc Maths and Physics	Bsc Physics
BSc Maths and Comp. Sci.	MMath	MMath Maths and Physics
MPhysics	PhD in Maths	PhD in Physics

Here extra empty columns were added to make more space between columns.

Instead of left-justified option l, one can use `p{<width>}`, which will wrap left-justified text with given width, useful if text is long. Example: using `\begin{tabular}{|l|r|p{4cm}|} ..... \end{tabular}`:

Maths	23	students who are all very clever and eager to learn maths
Physics	12	PhD students who are all very clever and eager to do research
Comp. Science	123	members of staff who are all highly qualified computer scientists

There are many other tricks for tables, how to make rows thicker, how to span rows or columns, how to align by the decimal point, etc., see manuals if interested.

Table as a 'float'. To have a table nicely (automatically) embedded in the text, and have number, caption, use
`\begin{table}[..]\caption{..}\label{..}\centering .. \end{table}`, where square brackets contain optional position 'wish' (t for 'top', b for 'bottom', h for 'here'):

Table 1: Listing

Maths	23	students
Physics	12	PhDs
Comp. Science	123 staff	mixed
Others		

Then we refer like "In Table 1 we have ..." by `In Table \ref{t1}...`

EXERCISE: Open 5fifth.tex, read, build (run build **twice!**), view. Save as my5fifth.tex. Open 5fifth-r.pdf. Writing over my5fifth.tex, produce input file such that output would match 5fifth-r.pdf. **Use environments, labels, and `\ref{..}` commands, not 'manual' numbers!** (Remember to run build **twice** to get cross-references right.) Recall that `\hline` can be used either at the beginning of tabular, or after `\\`.

4 Lists

Itemize. Write

```
\begin{itemize}
  \item The first item
 \item The second item ........
\end{itemize}
```

for your standard bulleted list of items:

- The first item The first item The first item The first item The first item The first item The first itemThe first item The first item The first item The first item The first item The first item
- The second item .....

In itemize one can replace bullets by anything using `\item[..]` .

Enumerate. The enumerate environment is for ordered lists, where by default, each item is numbered sequentially.

Nested lists. Lists can be nested:

```
\begin{enumerate}
  \item The first item
  \begin{enumerate}  (for 'inner' list)
    \item Nested item 1
    \item Nested item 2
  \end{enumerate}  (end of inner list)
  \item The second item
  \item The third etc \ldots
\end{enumerate}
```

produces

1. The first item
   (a) Nested item 1
   (b) Nested item 2
2. The second item
3. The third etc ...

EXERCISE: Open 6sixth.tex, read, build, view. Save as my6sixth.tex. Open 6sixth-r.pdf. Writing over my6sixth.tex, produce input file such that output would match 6sixth-r.pdf. **Use environments, not 'manual' numbers!**

5 Pictures

LATEX⇒DVI is tricky with pictures. Instead, change "Output profile" to LATEX⇒PDF (in drop-down menu). View the PDF output by Adobe Reader directly from the folder by double-clicking the PDF file (not by pressing "view output" in TEXNICCENTER).

(Actually one can make Adobe Reader working correctly from TEXNICCENTER in INB2305 by modifying the configuration: in menu "BUILD⇒DEFINE OUTPUT PROFILES⇒VIEWER" choose profile LATEX⇒PDF in the left pane, then in the three "server" boxes add without space R17 after each "acroview", thus replacing it by "acroviewR17". On the networked machines using PDF is fine, different PDF viewer is linked there.)

For images, top-matter should include `\usepackage{graphicx}`. Various types of images can be included, here JPG and PDF. The files for the pictures must be either in the same folder as the input file (easiest), or a path should be specified byu a special command in top-matter.

Command is `\includegraphics[width=0.1\textwidth]{constr-triangle.jpg}` to include the image `constr-triangle.jpg`, with width specified in square brackets. Best results as 'float' within `\begin{figure}[h] .... \end{figure}` environment:

`\begin{figure}[h]` (with optional [h] for 'here')
`\centering` (optional)
`\includegraphics[width=0.3\textwidth]{constr-triangle.jpg}\\`
`\caption{How to construct triangle}\label{pict1}` (all optional)
`\end{figure}`

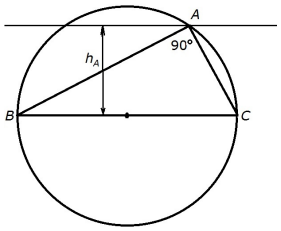

Figure 1: How to construct triangle

Then we can refer as "...see Fig. 1", by using `\ref{pict1}`.

EXERCISE: Open 7seventh.tex, read, build (twice), view. Save as my7seventh.tex. Open 7seventh-r.pdf. Writing over my7seventh.tex, produce input file such that output would match 7seventh-r.pdf. Use environments, not 'manual' numbers or captions!