LATEX INSTRUCTIONS

These instructions consist of 7 sections. For each section there are two files like 1first.tex and 1first-r.pdf, then 2second.tex and 2second-r.pdf, etc. You can learn the material of each section by reading the corresponding .tex file. In each section your exercise in class is to create your own .tex file which will produce .DVI output file similar to the given pdf file. After these in-class sessions on 31 October and 3 November, your LATEX assignment will be to create a .tex file that produces a DVI file with required features. Read the instructions how to edit .tex files, how to produce DVI files, etc.

Introduction

TEX (pronounced as [tekh] or simply [tek]) is a document-typesetting word-processing system, especially suitable for mathematics and physics.

There are various versions of TEX; the most popular is LATEX. LATEX is used in almost all mathematics and physics publications; all mathematical journals require submissions in LATEX; most maths books are published using LATEX.

Main difference from "What-you-see-is-what-you-get" (like WORD) is that LATEX works with

a) input (=source) tex-file like Filename.tex, which is a plain text file, where your text is combined with commands, essentially a program (code). These commands-instructions tell a special 'engine' how to process; the result of processing is a 'read-only'

b) output file Filename.dvi (can also be PDF — but we shall mainly work with DVI).

Synonyms: processing = building = typesetting = compiling = assembling.

The input file can be edited by any plain-text editor. Then the input text file is processed by an 'engine'. We shall use the most popular MIKTEX engine installed on these machines (and on the cloud).

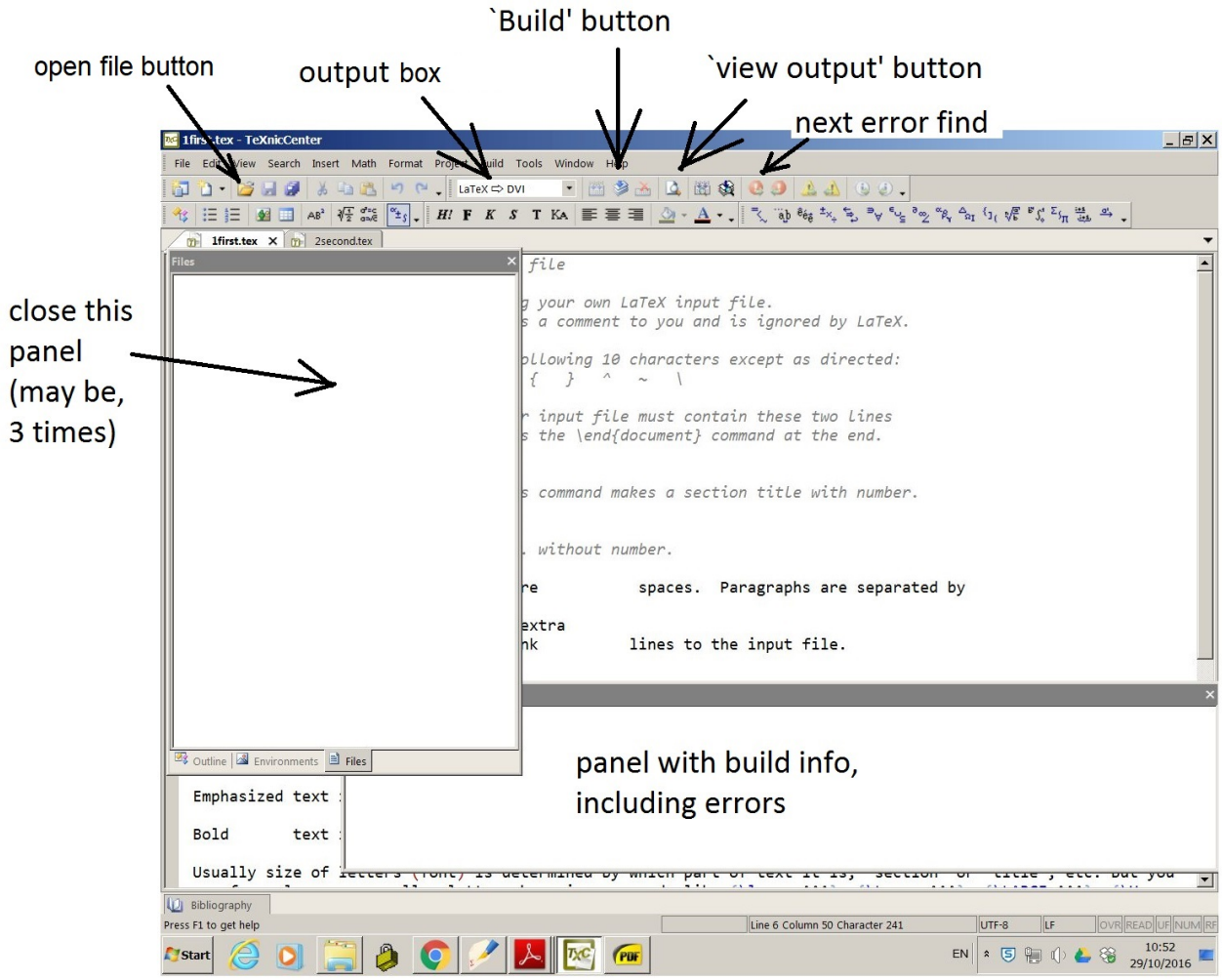
As an editor we use TEXNICCENTER installed on these machines (and on network). This editor already has in-built connection to MIKTEX, so we do not have to use MIKTEX commands ourselves. All work is done in TEXNICCENTER, the DVI output is viewed by DVI-viewer. Here we use the viewer YAP from MIKTEX.

(OPTIONAL: It is also possible to produce PDF files, which can be viewed by Adobe Reader. But when the PDF file is open in Adobe, we cannot process the input file, so one has to close PDF–Adobe first, each time. This can also be done automatically by a change of configuration in TEXNICCENTER on these machines: 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.)

Go to your BLACKBOARD, to "Professional Skills...", open "LATEX" folder in the "Learning materials"; download all the files (usually to ...USER⇒USER⇒DOWNLOADS). It may be more convenient to make a new folder LATEX and place all the files therein. **(RECALL: These machines are not networked, so no access to your H: drive; when finished, if you wish to save some files, use a memory stick, or send by email to yourselves, or save to the Cloud storage you have.)**

Open TEXNICCENTER from Aps tiles or "start menu". If used for first time, Config Wizard appears, press "NEXT", "NEXT" (to automatically link to MIKTEX), "NEXT" (no need to specify postscript options), "FINISH".

Make sure ‘Output profile’ box shows ‘LATEX ⇒ DVI — if not, choose in the drop-down menu. There may be layered left panels ”Outline”, ”Environments”, ”Files”, which can be closed.



1 Simplest document.

Open 1first.tex with TEXNICCENTER (either by double clicking on it, or from TEXNICCENTER ‘File’ menu, click ‘Open’, ‘File’, choose 1first.tex).

Read 1first.tex in TEXNICCENTER. You see a simple input file. Commands are in blue. Lines starting with % are comments/remarks, which are ignored by LATEX, just to clarify smthg when we work with editor.

The command `\section{..}` makes a section title with automatically generated number. The command `\section*{..}` makes a section without number. The command `\subsection{..}` makes a subsection with number. There is also `\subsubsection{..}`.

There are ten special symbols that cannot be typed as text: & $ # % _ { } ∧ ∼ \. These symbols are used as parts of commands:

$ is used to indicate maths; & for tabs in tables, etc.; _ for indices (=subscripts);

{ } to indicate bounds of commands; \ starts all commands, etc.

But if you really want these as characters in output, some (not all!) can be typed by adding \:

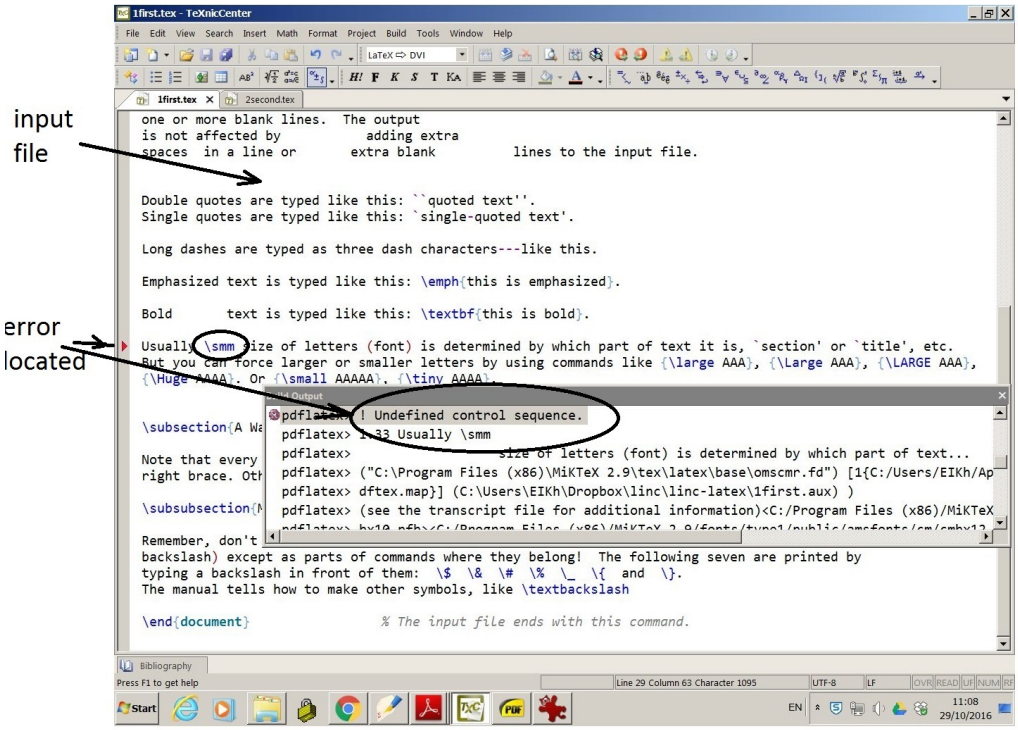
`\&` `\$` `\%` `\_` `\{` `\}`.

Further read 1first.tex about spaces, empty lines, bold, italics, large, small.

Now press 'Build current file' button (with blue down-arrow, second on the right of LATEX⇒DVI output profile' box). 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), make corrections, 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.

Settings of 'Build' can be different: as it is, building is tolerant, does not stop when errors occur, and usually some output is produced (if not too severe errors!). Try to corrupt `\section` like `\sectttion` and 'build'. You can still see output, but section is not right, red icon appeared, click 'next error' 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. Especially every left brace in commands (specifying the scope) like in `{\Large MMM}`, must be matched with right brace. Otherwise an error, or unintended result. E.g. try removing right brace in `\subsubsection{Memento}`, then try 'Build' – fatal error! even difficult to find... Restore this right brace! Try remove right brace in `{\Large MMM}`, then try 'Build' , 'View' – what happens? ... Restore the brace.



EXERCISE: Save 1first.tex as my1first.tex. Open 1first-r.pdf. Write over my1first so as to make output my1first.dvi similar to 1first-r.pdf ('reverse engineering'). Obviously, use select by mouse, cut-out CTR-x, copy and paste CTR-c and CTR-v for speed. It is usual to 'cannibalize' old LATEX input files, to save time by re-using many of the commands, like `\documentclass{article}`, `\begin{document}`, `\section{...}`, etc. Just remember to 'save as' with a different name!

2 More structure to a document.

Open 2second.tex.

After `\begin{document}`: see commands for Title, author, address, note the command `\maketitle`

- `\title{`*title*`}` - The title of the article.
- `\date` - The date. Use:
  - `\date{\today}` - to get the date that the document is typeset.
  - `\date{`*date*`}` - for a specific date.
  - `\date{}` - for no date.

`\begin{abstract}... \end{abstract}`

Author: the basic article class only provides the one command:

`\author` - The author of the document. Extra info like address in new lines, which are forced by \\

> **Remark:** For a slightly more logical approach, one can use the AMS article class: this means at the very top of input file replace `\documentclass{article}` with `\documentclass{amsart}`; then you have the following extra commands:
>
> - `\address` - The author's address. Use the new line command (\\) for line breaks.
> - `\thanks` - Where you put any acknowledgments.
> - `\email` - The author's email address.
> - `\urladdr` - The URL for the author's web page.

NUMBERING AND LABELLING. Many things are automatically numbered (like sections we saw), all numbered elements may be labelled:

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

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

This automatic cross-referencing is an important feature of LATEX. **It will be required in the assignment to use `\label{..}` and `\ref{..}` commands, rather than putting numbers 'manually'.** The build command must be run **twice** to make cross-referencing right.

BIBLIOGRAPHY: This a list of publications to which references are made in the text.

The special bibliography commands

```
\begin{thebibliogrpahy}
\bibitem{lab1} A. Hunter, Great Minds, \textit{J. Astronomy} {\bf 34} (2009), 23--45.
\end{thebibliogrpahy}
```

... creates reference with label lab1 for citing. Automatically numbers, and when referred to gives correct references like [1], [2], ... In the text cited by `\cite{lab1}` or `\cite[Theorem 4]{lab1}` (square brackets are optional containing an item from this publication).

EXERCISE: Save 2second.tex as my2second.tex. Open 2second-r.pdf. Write over my2second.tex so as to make the output my2second.dvi as in (similar to) 2second-r.pdf ('reverse engineering'). **Use labels! not 'manually' numbers for sections; use `\cite` commands, not simply type [1], [2].** Build. (Remember to run build **twice** to get cross-references right.) View. if errors occur, make corrections in the tex file in TEXNICCENTER if necessary, again build twice, view...

3 Mathematical formulae.

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

Mathematical formulae in LaTeX are between $ signs. E.g. $x+y=z-\alpha$ is produced by `$x+y=z-\alpha$`. (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.

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

`+ - = ! / ( ) [ ] < > | ' :`

NOTE: but curly braces are produced by `\{ .. \}`
while simply braces `{ .. }` are used in commands, to indicate scope, like `{\bf .. }`.

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

MOST COMMON CONSTRUCTS: (these examples are given without enclosing `$..$`, which however must be in the input file to produce formulae, it will not work without these $s!!.)

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

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

Indices (=subscripts): `_{}`, 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: $a^2_{ij}=a^2_{ij}$ 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}}}`.

Integrals: `\int`, like $\int f(x)dx$ from `\int f(x)dx`, or $\int_1^y(x^2+1)dx$ from `\int_{1}^{y}(x^{2}+1)dx`.

Displayed formulae: A big formula looks better in so-called displaystyle: between $$ ... $$: for example, the formulae above become

$$\frac{x-2}{(x^2-b)(a+b)} \qquad \sqrt[4]{\frac{a+b}{c+\sqrt{d}}} \qquad \int_1^y (x^2+1)dx.$$

If you want displaystyle but not displayed: use `\displaystyle{..}` command: $\displaystyle{\frac{x-2}{(x^2-b)(a+b)}}$ from `$\displaystyle{\frac{x-2}{(x^2-b)(a+b)}}$`.

Spaces in formulae: LaTeX does not recognize spaces in formulae. If needed, one can add `\quad` (as in the last displayed) (or smaller spaces `\,` or `\;`).

Brackets: Without them, formulae can become ambiguous. We already have the ( ) [ ] symbols 'from keyboard'. So why the need for more brackets? Example: 'normal' brackets may be too small, ugly:

$$(\frac{x^2}{y^3});$$

better use resizable brackets for large expressions:

$$\left(\frac{x^2}{y^3}\right)$$

from `$$\left(\frac{x^2}{y^3}\right)$$`.

The `\left...` and `\right...` commands provide automatic sizing of brackets. You must enclose the expression that you want in brackets with these commands. The dots after the command should be replaced with one of the characters depending on the style of bracket `(..)` or `[..]` that you want.

NOTE: remember, curly braces are only by `\{ or \}`! Example: $\{x$ such that $x^2 > 4\}$ by `$\{x$` such that `$x^2>4\}$`.

Greek letters: `\alpha`, `\beta`, `\gamma`, `\delta`, ... `\Omega`, `\omega` (see manuals for more).

Plain text within formula: The easiest is to use `\mbox{..}` – remember to add spaces if needed. Example: $\{x$ such that $x^2 > 9\}$ is produced by `$\{x\mbox{ such that } x^2>9\}$`.

Unions and intersections: $A \cup B$, $A \cap B$ produced by `A\cup B and A\cap B`.

In displaystyle better use big ones:

$$\bigcup_{i=1}^{n} B_i, \qquad \bigcap_{j=1}^{m} D_j,$$

where `\bigcup` and `\bigcap` are used.

Product, sums: Summation sign $\sum$ is given by `\sum`. Product sign $\prod$ is given by `\prod`. Can have indices:

$$\sum_{i=1}^{n} a_i = a_1 + a_2 + \cdots + a_n;$$

is produced by `\sum_{i=1}^{n}a_i=a_1+a_2+\cdots +a_n`.

Product:

$$\prod_{i=1}^{m} b_i = b_1 \cdot b_2 \cdots b_m;$$

is produced by `\prod_{i=1}^{m}b_i=b_1\cdot b_2\cdots b_m`. Note **nice dots** given by `\cdots`. Another version of dots is `\ldots`, which gives like $a_1, a_2, \ldots, a_n$ from `a_1,a_2,\ldots ,a_n`.

Infinity symbol: ∞ is given by `\infty`. E.g.

$$\sum_{i=1}^{\infty} b_i = b_1 + b_2 + \cdots$$

is produced by `\sum_{i=1}^{\infty }b_i=b_1+ b_2+\cdots` .

EXERCISE: Open 3third.tex. Read about formulae. Build. View the result. **Save 3third.tex as my3third.tex. Open 3third-r.pdf. Reverse-write an input file by writing over my3third.** Hint: for convenience do not delete the old text, but write before it, after `\maketitle`, and look-up in the old text and copy any construction that you need; delete old text only after build–view iterations succeed without errors.

4 Environments.

Open and read 4fourth.tex. 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{..}` (which you choose to be unique for every equation, theorem, etc.).

EQUATION ENVIRONMENT:

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

produces the numbered equation

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

Then we can refer to it by "In equation (1)..." by using `(\ref{lbl})` (referring to that label). 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.

> OPTIONAL REMARK: 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' after `\documentclass{article}` and before `\begin{document}`:

`\newtheorem{thrm}{Theorem}`,

where first brace `{thrm}` is the name (you choose) of this theorem-type environment and second `{Theorem}` tells LATEX what you choose to type 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:

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

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

We saw in Lemma 1 that... (using `Lemma \ref{l1}`).

Proof: if using `\usepackage{amsthm}` (can be added right after `\documentclass{article}`), 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 dvi 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.)

5 Tables.

Read 5fifth.tex. 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		

from

```
\begin{tabular}{|r|cl|}
\hline
Maths & 23 & students\\
Physics & 12 & PhDs\\
Comp. Science & 123 staff & mixed\\
\hline\hline
Others & & \\
\hline
\end{tabular}
```

Note that the special symbol & must appear the same number of times for each row!

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

from

```
\begin{tabular}{l c l c l}
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
\end{tabular}
```

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

from (see next page)

```
\begin{tabular}{|l|rp{4cm}|}
\hline
Maths & 23 & students who are all very clever and eager to learn maths\\
\hline
Physics & 12 & PhD students who are all very clever and eager to do research \\
\hline
Comp. Science & 123 & members of staff who are all highly qualified
computer scientists
\hline
\end{tabular}
```

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 * in square brackets denotes one of three optional position 'wishes': t for 'top', b for 'bottom', h for 'here':

Table 1: Listing

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

from

```
\begin{table}[h]\caption{Listing}\label{t1}
\centering
\begin{tabular}{|r|cl|}
\hline
Maths & 23 & students\\
Physics & 12 & PhDs\\
Comp. Science & 123 staff & mixed\\
\hline\hline
Others & & \\
\hline
\end{tabular}\end{table}
```

Note: `\label` after `\caption`.

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, `\label`, 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 `\\`.

6 Lists

Read 6sixth.tex.

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 first itemThe 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. Write

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

to produce

1. The first item
2. The second item .....

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!**

7 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.)

Read 7seventh.tex. 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 by 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. But better use environment 'figure' given by `\begin{figure}[h] .... \end{figure}`, for example:

`\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}`

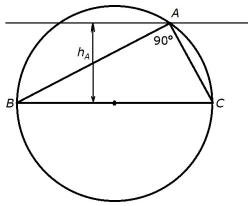


Figure 1: How to construct triangle

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

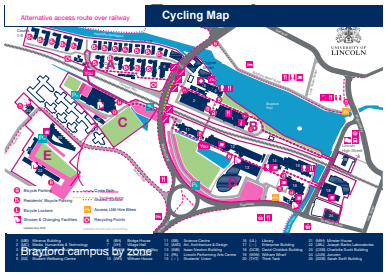


Figure 2: Campus map

EXERCISE: Open 7seventh.tex, read, build (twice), view. Save as my7seventh.tex. Open 7seventh-r.pdf. Writing over myseventh.tex, produce input file such that output would match 7seven-r.pdf. Use environments, `\label`, `\ref` commands, not 'manual' numbers; use `\caption`, not 'manually'!