Course B
Large Documents

There are many considerations that go into creating a paper or thesis. We will touch on the major ones here, including

- Front matter
- Table of contents
- Enumerating sections, subsections, sub-subsections, sub-sub-subsections, etc.
- Enumerating definitions, theorems, lemmas, etc.
- Bibliography and citation management.

This last item is probably more important (and problematic!) than you realize. With a little bit of set-up time, you will never have to think about bibliographic organization ever again.

Basic Structure and Layout

No matter what you're doing, the general appearance of any document is controlled by the `\documentclass` declaration. By "general appearance" I mean things like the typeface of the title line and other front matter, the style of the chapter/section/subsection titles, the way that equations are tagged, the style of the theorems, definitions, etc., and other structural components.

For the most part, one never worries about such things until you have to. More than likely, the document class `article` will be all you ever need.

WARNING: I'm going to be using Grätzer's examples today. These examples contain all the bells and whistles, many of which you will never need. Thus we will be ignoring some parts of the code. Also, he uses the document class `amsart` instead of `article`. Just be aware.

Front Matter

Skipping the header for now, open up the file `sampart.tex` from the samples directory. We're really going to mess this thing up, so save it under a different name. The bit right after the `\begin{document}` line contains all the front matter information: title, author, date, abstract, etc. The `\maketitle` command takes the information specified in the fields and sets it *according to the specifications mandated by the document class*. If your document class does not use certain fields, LaTeX will either ignore them or give an error (most likely the former).

All of this is pretty straightforward, but there are two things worth mentioning. First, you'll notice that the `\title` command seems to have two arguments. The actual title goes in the curly braces. Some document classes run the title at the head of every other page. In the event that your title is too long, you'll want to give an abbreviated version for the running header. This is what goes inside hard brackets. So the general format is

```
\title[My Short Title]{The Actual Long-Winded Title of My Paper Goes Here}
```

The other thing worth knowing about is *unbreakable space*. Sometimes you do not want to allow LaTeX to break a phrase across a line. For example: a result called "Theorem 3.2" should be typeset as a single object, and a line should not be allowed to end with "Theorem" and the next line start with "3.2." This is what tildes are for. By typing `Theorem~3.2` LaTeX will bind the two parts together, preventing a break across lines. Note that George has done this with his middle initial to keep it from bleeding down into the next line (if there would ever be one).

Exercise #1. Let's do these now.

1. Steal George's article: put your name on it.
2. Change the running header title to "I Stole This!"

The Table of Contents
Well, I could go on and on about how to create a table of contents, but I won't. It can get really complicated. If you just want a basic table of contents without any thinking, just place

```
\tableofcontents{\protect\thispagestyle{empty}}
```

right before the first text section of your paper. This will take all your sections and subsections, determine their numbers, determine which page they're on, then create the table of contents. The point of the line `\protect\thispagestyle{empty}` is to suppress the page numbering on the first page, which is somewhat traditional. If you actually want that page number back, just get rid of that argument.

Naturally, the appearance of the table of contents is a function of the document class!

Exercise #2.

1. Give "your" paper a table of contents, suppressing the numbering on the first page.
2. Now give "your" paper a "title page" with no text on the first page, just the front matter and the table of contents.

NOTE: Whenever you shift pages or reference labels, it will take LaTeX several runs to assemble all the changes you have made. When in doubt, compile twice in a row. More on this hassle later.

Enumerated (Sub)sections
This is really simple. To create a new section with a reference label, just type

```
\section{My Section Name}\label{my label name}
```

LaTeX will start a new section with the heading "My Section Name." It will automatically assign it a section number in sequence. *The numbering scheme and style is a function of the document class.* In articles, sections typically enumerate as 1, 2, 3, ... with subsections 1.1, 1.2, 1.3, ... In a document class like `book` or `report` you have chapters before sections (so chapter 1 will contain sections 1.1, 1.2, ... and chapter 2 will contain sections 2.1, 2.2, ...).

Any of these subdivisions may be labelled dynamically, working in a very similar manner to last week's discussion about labelling equations. The only difference is that references to labels are made by `\ref{my label name}` instead of `\eqref` (after all, we're not referencing equations). Again, upon inserting or deleting labelled matter, LaTeX may need to compile several times in order to assimilate all the changes.

Exercise #3.

1. Give "your" paper an extra section between sections 1 and 2. Call it "This Is a Fake Section." Don't forget to label it!
2. Now refer to your fake section at the end of the paper. Don't cheat!

Enumerating Theorems, Definitions, Etc.
We've all seen this in mathematics textbooks: Theorem 2.3, Definition 7.2, etc. As you have probably already guessed, LaTeX can automatically enumerate these proclamations for you. This can get rather complicated, for two reasons:

- You have to create the proclamation environments yourself, and
- You have to decide how to sequence different types of proclamations.

For example, do you need lemmas or not? If you do, you have to tell LaTeX this ahead of time. Also, do you want sequencing like

Theorem 1, Definition 2, Lemma 3, Theorem 4, Definition 5

or would you prefer

Theorem 1, Definition 1, Lemma 1, Theorem 2, Definition 2?

Apparently George prefers the latter, but that's his business. Let's examine his code:

```
\theoremstyle{plain}
\newtheorem{theorem}{Theorem}
\newtheorem{corollary}{Corollary}
\newtheorem*{main}{Main~Theorem}
\newtheorem{lemma}{Lemma}
\newtheorem{proposition}{Proposition}
```

The first line dictates that all of the following proclamations will be typeset in the `plain` theorem style. This essentially means "use italics," but *the complete layout is a function of the document class.*

Next, he types `\newtheorem{theorem}{Theorem}`. This means that he is creating an environment called `theorem` which will typeset the word "Theorem" (with the appropriate sequence tag!) when invoked. So when George is ready to state a theorem, he codes `\begin{theorem}`, types his theorem, then closes with `\end{theorem}`. Thus the basic format for a proclamation declaration is

```
\newtheorem{command name}{Typeset Name}
```

This is the most basic kind of declaration, with none of the options. This gives the kind of sequencing that George likes (the second example above). And one more thing: when George is ready for a proof, he codes `\begin{proof}` and ends it, well, you know how. LaTeX handles the spacing and closing of the proof automatically.

Next, you'll notice that George typed the following in his header:

```
\theoremstyle{definition}
\newtheorem{definition}{Definition}
```

He is now changing the style! The first line means that every proclamation that follows will be typeset in the `definition` style, which essentially means "no italics." This does *not* affect the previous declarations. There are three basics styles for proclamations: plain, definition, and remark. Consult a good manual for more details.

So how do you change the sequencing? For instance, as a matter of personal taste I prefer everything to be numbered in the same sequence, not four different ones. To do this, you use an optional argument right after the command name in the declaration. For example, to get your lemmas to be numbered in the same sequence as your theorems, code

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

The argument `[theorem]` has to be a *previously-defined command name.*

Exercise #4.

1. Define "your" lemmas and corollaries so that they're in sequence with "your" theorems.
2. Now change the `\newtheorem{theorem}{Theorem}` line to `\newtheorem{theorem}{Theorem}[section]`. Guess what it will do before compiling it.

Implementing BibTEX

BibTEX is an exceptionally strong package that allows you to manage your bibliography and the citations dynamically, with very little effort. Using it the first time or two will result in some headaches, but after that it's very simple to use and maintain.

Using BibTEX requires two things: a database and a "style file." The database consists of the bibliographic data for the referenced items, and must always end with the `.bib` extension (and, as usual, no spaces allowed). The style file tells LATEX how to take the database entries and format them into the document, and they always end in the `.bst` extension. The standard style files probably came with your LATEX bundle, but there are hundreds of others out there. All of today's examples will be in the `amsplain` style.

Let's look at George's database (his "bib file"). Open the document `sampartb.bib` from the samples directory. Look, but don't touch. I feel that the format for entering the data is fairly clear, but there are a few things to point out:

- A bib file has no header. It consists of data alone.
- Each new bib entry starts with an `@` sign.
- The data for an entry starts with `(` and ends with `)`. You can also use curly braces.
- The first argument following the "type" declaration is the label for that entry.
- Fields are separated by commas.
- The data in a field is enclosed by quotation marks.
- If you want to protect capitalization, use curly braces.

If you want a complete list of all possible types (`book`, `article`, `phdthesis`, etc.) and fields, consult a good manual. Also, some fields may not be used by your particular style file. If a particular style does not use a field, it will simply ignore it without giving an error.

Assuming you have a database, how do you use it? First and foremost, *your bib file MUST be in the same folder as your source code.* If you want to cite an article in your source code, just call it up with `\cite{label name}`.

Sometimes you want to include an article in your bibliography even though it wasn't cited in your paper. In this case, coding `\nocite{label name}` will add the entry to the bibliography, even though you never used it. See, you have to do this because BibTEX will only add the entries from the database that you actually cited.

Next, we have to make LaTeX work with BibTeX to get the final product. This is not too bad, but it's not exactly simple either.

Let's open up `sampartb.tex`, which is simply George's article with BibTeX added. Look at the very end of his article. There you'll see

```
\bibliographystyle{amsplain}
\bibliography{sampartb}
```

The first line tells LaTeX which bibliography style file to use. The second line tells it which database to use. Note that neither of these use the `.bst` or `.bib` extension.

We're now ready to compile the final product. Because there is so much information for LaTeX to assimilate and self-reference, you must compile by the following recipe:

$$\text{LaTeX} \Rightarrow \text{BibTeX} \Rightarrow \text{LaTeX} \Rightarrow \text{LaTeX}.$$

(It's better not to ask about this, just take it on faith.) If you have made edits that do *not* affect the bibliography in any way, you can "single-compile" as usual. But whenever you make changes that affect the citations, you must follow the above 4-step-compile process to get the corrected new version. Sorry, that's life.

Internal Bibliographies: It is certainly possible to create a nice bibliography *without* using BibTeX. If you look at the source for George's original article (`sampart.tex`), you'll see that all the bibliography information has been added manually at the end of the code. This works, but it's also a lot of work on your part to get everything in the "right" format.