A PRACTICAL INTRODUCTION
TO TEX AND LATEX

James Holland

2021-09-30

§ 0 A. What exactly is 'TEX'?

TEX is a way of marking plain text to signal how a document should be formatted. To give some historical context, if you were to write a book, the publisher would be in charge of the font choices, spacing, etc. You would merely say things like "hey, this is the start of a new chapter" or "hey, emphasize this phrase" by marking the text accordingly, and the publisher would format things as they desired. TEX is basically the notation used (by-and-large) in math for markings like this.

Ordinarily, typesetting mathematics can be done one of three ways.

- Using TEX or LATEX.
- Using an equation editor built into a program like Microsoft Word or Google Docs that mimics the functionality and syntax of TEX.
- Approximating the written notation using obscure unicode characters, e.g. approximating "$\int_1^a \frac{1}{x^2+1}\,\mathrm{d}x$" with

  "∫_1^a 1/(x^2+1) dx"

  or an even more complicated but slightly better looking

  "∫₁ᵃ 1/(x²+1) dx".

One of the benefits of TEX and LATEX is being able to produce math expressions with nothing more than what's found on a standard keyboard, allowing even more complicated expressions where simple unicode replacements would be insufficient, e.g.

$$\text{“}f_1^*(\vec{x}_\alpha) = \sum_{n=1}^{2^{10}} \left(\frac{\delta_n'(\ddot{x}_n, \ddot{y}_n)}{\int_0^{33+n} 2^n - \frac{x}{n}\,\mathrm{d}x} \right)^2\text{”}.$$

LATEX is an expansion of TEX that basically just standardizes some things and adds some common commands. In this way, TEX is not like Word or Google Docs where "what you see is what you get". TEX is more like a computer language, and in principle can be written on a napkin.

Turning TEX into an actual document, like .pdf or .djvu etc, requires a separate program that inputs the plain text and outputs a well-formatted document. In addition to such a program, it's also common to have a program that's able to import new notation, commands, etc.—*packages*—that other people have defined. Often these two types of software are installed together. Common programs include

1. MiKTeX (the package organizer/distributor) with TeXworks (the document creator),
2. TeX Live (the package organizer/distributor) with TeXworks (the document creator),
3. MacTeX (essentially a modification of TeX Live) with TeXShop (the document creator),
4. TeXstudio (the document creator),
5. Overleaf (a website doing both package organizing/distributing and document creation online),
6. and many more.

Package distributors/organizers usually simply import packages from Comprehensive TEX Archive Network (CTAN) which you can download from directly, but it's kind of a hassle to deal with the packages yourself. You will almost certainly be using packages, so using one of the above distributions is highly recommended.

Generally speaking, it's easiest to simply use Overleaf until you want to be able to better organize your files. Moreover, using Overleaf allows you to pretty easily experiment without worrying that you screwed up the installation of other software on your computer or anything.

Section 1. *Creating a Document Example*

Below is an example of a very basic homework solution written in TEX, followed by the actual pdf it generates. Let's walk through what all of this means and why it's written like it is.

```
\documentclass{article}
\usepackage{amsmath}
\usepackage{amssymb}

\author{A. Person}
\title{My Homework}
\date{January 1, 1900}

\begin{document}

        \maketitle

        \textit{Problem 1: Show that if \(A\subseteq B\) and \(B\subseteq C\) then \(A\
            subseteq C\).}
        \vspace{10pt}

        \textbf{Solution:} Suppose that \(x\in A\) is arbitrary.  By the hypothesis
            that \(A\subseteq B\), it follows that \(x\in B\).  By the hypothesis that
            \(B\subseteq C\), we similarly have that \(x\in C\).  As \(x\in A\) was
            arbitrary, this shows that \(A\subseteq C\).
        \vspace{.5in}

        \textit{Problem 2: Show that the set
        \[A=\{x\in\mathbb{R}:x^2=-1\text{ or }x+1\in\mathbb{N}\}\]
        is non-empty.}
        \vspace{10pt}

        \textbf{Solution:} It suffies to give an example of an element in \(A\).  It's
            easy to see that \(0\in A\) since \(0+1=1\in\mathbb{N}\).

\end{document}
```

1 • 1. Figure: A Simple Homework Written in TEX

This generates the following. There are of course some things that can be improved here—e.g. the margins—but there are many things to explain first.

My Homework

A. Person

January 1, 1900

Problem 1: Show that if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.

Solution: Suppose that $x \in A$ is arbitrary. By the hypothesis that $A \subseteq B$, it follows that $x \in B$. By the hypothesis that $B \subseteq C$, we similarly have that $x \in C$. As $x \in A$ was arbitrary, this shows that $A \subseteq C$.

Problem 2: Show that the set

$$A = \{x \in \mathbb{R} : x^2 = -1 \text{ or } x + 1 \in \mathbb{N}\}$$

is non-empty.

Solution: It suffies to give an example of an element in A. It's easy to see that $0 \in A$ since $0 + 1 = 1 \in \mathbb{N}$.

The symbol \ indicates that what follows is a command. For example, \maketitle tells the computer to print out the title, author, and date of the document and format them appropriately. How does it know what is "appropriate"? This is determined by the *document class*. TEX can be used to create books, articles, letters, slide shows, and many many more kinds of documents that have their own style conventions. The document class just makes these sorts of things explicit. For now, the article class article will suffice for us, but standard document classes include book, report, memoir, beamer, letter, and minimal.

Frequently, commands include inputs, much like functions. The inputs of a command are surrounded by braces, e.g. the "input" in: \command{input}. For example,

- \documentclass{article} tells the computer to use the document class article.
- \author{A. Person} tells the computer to save the string of text "A. Person" as the author of the document.
- \vspace{10pt} tells the computer to create vertical space of 10 points (a *point* is a unit of measure, equal to 1/72.27 inches or about .35 millimeters).
- \textit{blah} tells the computer to write out "blah" in italics: *blah*.
- \mathbb{R} tells the computer to write out the letter 'R' in a blackboard bold font: $\mathbb{R}$.

In general, to group things together in TEX, one uses braces { and }.

And then there's one of the most common commands:[i] \(coupled with \). This indicates that you are writing a math expression using math symbols. For example, the text x<a yields "x<a" normally in TEX, but \(x<a\) gives the nicer "$x < a$".

Often many commands can only be issued in this so-called "math mode" (when writing math, i.e. between these \(, \)). For example, \mathbb{...} can only be used in math mode. Similarly, subscripts x_0 with an underscore and superscripts x^0 with a caret can only be used in math expressions (i.e. in math mode).

The last main topic parts of a TEX file are environments. These are enclosed with \begin{...} and \end{...}. Similar to how \(enters math mode and \) exits it, \begin{...} enters some new context where things may act differently compared to the rest of the text, either through formatting or commands. For example, \begin{document} tells the computer when to actually start writing things out in the document as opposed to, for example, collecting information like what packages to use and what the author's name is.

Let's now go line by line through Figure 1 • 1 to understand how you could create this document.

§ 1 A. The Preamble of Figure 1 • 1

- The first step is to specify the document class. article works for most purposes with homework. So we add the line

```
\documentclass{article}
```

- Now we should include some basic packages that add some fonts so we can use symbols like $\mathbb{R}$. So we add the packages amsmath and amssymb by $\mathcal{AMS}$. So we add the lines

```
\usepackage{amsmath}
\usepackage{amssymb}
```

- Now we specify just what exactly this document is supposed to be: who's writing it, and what's it called? This gives the lines

```
\author{A. Person}
\title{My Homework}
```

- For good measure, we can optionally include the date. Generally speaking, to get today's date, we can just put \date{\today}. To get a different date, we can just put in whatever we want, in whatever format we want. So we could add in the following line as in Figure 1 • 1.

[i] Some people write $ for both \(and \) as in $x<a$. This is the original notation of TEX, but is less flexible than the LATEX \(and \). Either

```
\date{January 1, 1900}
```

- Now we have all of the *preamble* set, and we can start actually writing the document. To mark the start of the document, we begin the `document` environment with the line

```
\begin{document}
```

With the beginning of the document comes the end of the preamble.

§ 1 B. The document of Figure 1 • 1

- Firstly, let's write out the title and so on. We generally do this with a command like `\maketitle`, so we include that as a separate line.
- Now we must write the problems and their solutions. To distinguish the problems, let's write them in italics. This means we must use the command `\textit` where the input is the entire problem statement:

```
        \textit{Problem 1: ...}
```

- The problem statement contains some math and math symbols like '⊆'. How do we write this? Well, the easiest way to find a symbol is simply to look it up on the internet, especially by drawing it out on a website like Detexify, which will give the command to produce the symbol. Alternatively, one can look up the symbol in The Comprehensive LaTeX Symbol List. In either case, '⊆' can be produced with `\subseteq`. But remember that we must be in math mode! So to produce "$A \subseteq B$", we must surround the text to indicate that it is a math expression: `\(A\subseteq B\)`. Note that there is a space between `\subseteq` and `B` there: otherwise TeX will interpret it as the single command `\subseteqB`. Since there is no such command defined (usually), this would produce an error. So we then add the line

```
        \textit{Problem 1: Show that if \(A\subseteq B\) and \(B\subseteq C\) then
            \(A\subseteq C\).}
```

- Now we must format our solution. If we just go straight into it, there will be no space between the problem statement and the solution. To add such a space, we can use the command `\vspace{...}` and add however much space we think is appropriate. In Figure 1 • 1, I've used a 10pt space, but one could in principle use any length like 1 inch (`1in`) or 5 millimeters (`5mm`) or .5 centimeters (`.5cm`) and so on. So we include the following line, just for the sake of appearance.

```
        \vspace{10pt}
```

Now we have a subtle point. We must create a new, blank line beneath the command `\vspace{...}`. The reason is that TeX only has a paragraph end if there is an entirely blank line. If we were to not include a blank line, the space would be created after the above line ends. For example, consider the following

```
This is a sentence with words blah blah blah.
\vspace{10pt}
This has some more words too bleh bleh bleh.  Here have even more words blah blah blah.
     Those were some words, huh? Bleh bleh bleh.  Well, that's enough words for now
   blah blah blah.
```

Compared to including a blank line as in

```
This is a sentence with words blah blah blah.
\vspace{10pt}

This has some more words too bleh bleh bleh.  Here have even more words blah blah blah.
     Those were some words, huh? Bleh bleh bleh.  Well, that's enough words for now
   blah blah blah.
```

With no extra blank line, we get the following: LaTeX has formatted the text as a single paragraph with a weird space after the first line.

suffice though.

This is a sentence with words blah blah blah. This has some more words too bleh bleh bleh. Here have even more words blah blah blah. Those were some words, huh? Bleh bleh bleh. Well, that's enough words for now blah blah blah.

With a blank line in the LaTeX code, we get this.

This is a sentence with words blah blah blah.

This has some more words too bleh bleh bleh. Here have even more words blah blah blah. Those were some words, huh? Bleh bleh bleh. Well, that's enough words for now blah blah blah.

This properly includes space where we want it.

- Returning to Figure 1 • 1, this means we should place a blank line just after `vspace{10pt}`.
- Now we format our solution. To distinguish it from the problem statement, let's indicate that this is a solution by writing "solution" in bold. To do this, we use the command `\textbf` just as we did `\textit`. So we start out with

```
\textbf{Solution:} ...
```

- The rest of our solution uses the same techniques as before. How do we know how to produce "∈"? Again, we look it up using either Detexify, or The Comprehensive LaTeX Symbol List. Either way, `\in` produces ∈ in math mode.

```
\textbf{Solution:} Suppose that \(x\in A\) is arbitrary.  By the
    hypothesis that \(A\subseteq B\), it follows that \(x\in B\).  By the
    hypothesis that \(B\subseteq C\), we similarly have that \(x\in C\).
    As \(x\in A\) was arbitrary, this shows that \(A\subseteq C\).
```

- Now that we've completed our solution, let's create some more space to separate the problems. Again, we need an extra line afterward for LaTeX to understand where to place the space. And again, the half inch used here is just an arbitrary number that looks decent.

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

- Now we move on to the next problem, using the same principles as before. The only question is now, how did we get the math expression $A = \{x \in \mathbb{R} : x^2 = -1 \text{ or } x + 1 \in \mathbb{R}\}$ on its own separate line?
  - TeX has a *display* style for math expressions, used primarily for either long expressions, or important ones. To create these, rather than parantheses, use brackets: `\[` and `\]` will produce display math while `\(` and `\)` will produce *in-line* math (i.e. math expressions in a paragraph).
  - To write the superscript 2, we use a caret: `^{...}`. So we could write, for example, `x^2`, `x^{1+1}`, `x^{-1}`, and so on for superscripts, again using braces `{` and `}` to group things together. In this way, the caret is a command with no `\`.
  - But because `{` and `}` are used to group things together, how did we write the braces in math mode? One can look up the symbols online, but it's simply done with the commands `\{` and `\}`.
  - To write out text in math mode, we use the command `\text{...}`. For example, `\(A or B\)` produces "$AorB$": the "or" isn't formatted like text. However, `A\text{ or }B\)` produces "A or B" which is formatted correctly.

  So we get the lines

```
        \textit{Problem 2: Show that the set
        \[A=\{x\in\mathbb{R}:x^2=-1\text{ or }x+1\in\mathbb{N}\}\]
        is non-empty.}
```

- Now we can add another space between the problem and its solution, and format the solution as before.

```
        \vspace{10pt}

        \textbf{Solution:} It suffies to give an example of an element in \(A\).
            It's easy to see that \(0\in A\) since \(0+1=1\in\mathbb{N}\).
```

- This marks the end of the document, so we must tell the computer that we are done: there is nothing more after the document ends that it must do.

```
\end{document}
```

This explains all of Figure 1 • 1 and the document it produces, allowing you to create your own documents that look similar. While this may be a lot to take in at first, playing around with LaTeX can be fun, and overall, the look of your documents will improve.

Section 2. A More Complicated Example

Sometimes, you may not like how Figure 1 • 1 is formatted: the margins are too large, or the font is too small or big. You may not like constantly having to write `\vspace{10pt}` a bunch of times, and worse yet, if you decide to change that 10 pt to a different length, then you have to change it many, many times. You may want to introduce short-cuts or create your own commands or environments, or write comments that aren't processed by the actual document.

Let's consider a slightly more complicated example of a homework submission that complicates the preamble, but streamlines the writing of solutions. Let's first just consider the preamble: everything before `\begin{document}`.

```
\documentclass[letter]{article}
\usepackage{amsmath, amssymb}
\usepackage[margin=1in]{geometry}

\title{My New Homework}
\author{A. Person}
\date{\today}

\newenvironment{problem}[1]
        {       \noindent \textbf{Problem #1} \itshape  }
        {       \vspace{10pt}

        }
\newenvironment{solution}
        {       \noindent \textbf{Solution:}    }
        {       \vspace{20pt}
                \hrule
                \vspace{20pt}
        }
\newcommand{\sse}[2]{#1\subseteq #2}
```

2 • 1. Figure: A Second Homework Preamble Written in TeX

The changes from Figure 1 • 1 include the following (beyond the title change).

- The option `letter` for the document class, which specifies that we're using letter sized paper (this makes it easier to print in the United States where letter paper is the standard size).
- The new package `geometry` which makes it very easy to change margins to just one inch. This is done using the package option `margin=1in`. In general, using options for a package will have you writing

```
\usepackage[option]{package name}
```

- The date was changed to today (the date the .tex file was processed).
- We have defined two new environments: `problem` and `solution`. The idea is that we will put the problem statement in the `problem` environment and our solution inside the `solution` environment. But we still need to explain what we have written here.
- Added a new command that inputs two arguments and outputs that one is a subset of the other.

In general, an environment is really just two sequences of commands: a bunch of commands at the start of the environment, and then a bunch of commands at the end of the environment. We can also decide to include arguments or not. Generally, a new environment takes the following form.

```
\newenvironment{NewEnvi}[number of inputs]
        {commands executed at "\begin{NewEnvi}"}
        {commands executed at "\end{NewEnvi}"}
```

So for the new environment `problem`, we have decided to

1. Include one input by writing `[1]`. This input can be referenced with `#1` (if there were three inputs, we could reference the third with `#3` and similarly with the others).
2. Not indent the new paragraph by using `\noindent`.
3. Write out the text "**Problem #1**" where `#1` is supposed to be the problem number that the user is supposed to have input.
4. Set the rest of the text in the environment to be in italics with the command `\itshape`. We can't use `\textit` here because that requires the input to be enclosed with `{` and `}`, which will merely confuse the a LaTeX compiler with syntax errors. (Where does the environment definition begin and end?)

That all happens at the start of the environment. Users are then free to write whatever garbage they would like in the environment. Then at the end of the environment, we have decided to

1. Add a vertical space with `\vspace{10pt}`.
2. Start a new paragraph.

Similarly with the new environment `solution` that we defined with

```
\newenvironment{solution}
        {       \noindent \textbf{Solution:}    }
        {       \vspace{20pt}
                \hrule
                \vspace{20pt}
        }
```

We have decided to

1. Include no inputs.
2. Prevent the indentation of a new paragraph.
3. Write out "**Solution:**".

That all happens at `\begin{solution}`. Then at `\end{solution}`, we have decided to

1. Add space with `\vspace{20pt}`.
2. Add a horizontal line to separate the solution from the next problem using `\hrule`.
3. Add more space with `\vspace{20pt}`.

The command we added with

```
\newcommand{\sse}[2]{#1\subseteq #2}
```

Says that we have defined the command `\sse` to input two arguments. In action, `\(\sse{A}{B}\)` outputs "$A \subseteq B$". So each input should be enclosed in parantheses. In general, defining a new command takes the form

```
\newcommand{command name}[number of inputs]{what the command does}
```

Sometimes you don't want any inputs, in which case, one can write merely

```
\newcommand{command name}{what the command does}
```

In action, here is the rest of a .tex document (with the above preamble) using these environments and new command.

```
\begin{document}
        \maketitle

        \begin{problem}{1}
                Show that if \(\sse{A}{B}\) and \(\sse{B}{C}\) then \(\sse{A}{C}\).
        \end{problem}
        \begin{solution}
                Blah blah blah blah blah blah blah blah blah blah blah blah blah blah
                    blah blah blah blah blah blah blah blah blah blah blah blah blah
                    blah blah blah blah blah blah blah blah blah blah blah blah blah
                    blah blah blah blah blah blah blah blah blah blah blah blah blah
                    blah blah
        \end{solution}

        \begin{problem}{3.2}
                Show that the function \(f:\mathbb{R}\rightarrow\mathbb{R}\) defined by
                \[f(x)=\begin{cases}
                        x+1&\text{if }x<0\\
                        2x&\text{if }x\ge 0
                \end{cases}\]
                is not injective, but is surjective.
        \end{problem}
        \begin{solution}
                \(f\) is not injective because \(f(-1/2)=1/2=f(1/4)\).  To see that \(f
                    \) is surjective, let \(y\in\mathbb{R}\) be arbitrary.  If \(y\ge
                    0\), then \(y/2\ge 0\) and hence \(x=y/2\) gives \(f(x)=2x=y\).  If
                     \(y<0\), then \(y-1<0\) so that \(x=y-1\) gives \(f(x)=x+1=y\).
                    Since \(y\in\mathbb{R}\) was arbitrary, \(f\) is surjective.
        \end{solution}

        \begin{problem}{3.4(a)}
                Which of the following is a true statement?
                \begin{enumerate}
                        \item If \(f:A\rightarrow B\) is a function, then \(f\) is
                            always surjective onto its range.
                        \item If \(f:A\rightarrow B\) is a function, then \(f\) is
                            always surjective onto \(B\).
                        \item If \(f:A\rightarrow B\) is a function and \(\sse{B}{C}\),
                             then \(f\) is a function from \(A\) to \(C\).
                        \item If \(R\subseteq A\times B\) is a relation, then the
                            domain of \(R\) is \(A\) and the range of \(R\) is \(B\).
                        \item Every relation \(R\) has a subset \(f\subseteq R\) such
                            that \(f\) is a function.
                \end{enumerate}
        \end{problem}
        \begin{solution}
                \begin{enumerate}
                        \item True by definition of the range of \(f\): the range of \(
                            f\) is \(\{y\in B:\exists x\in A\ (f(x)=y)\}\).
                        \item False: consider \(f=\{(0,0)\}\) the function \(f:\{0\}\
                            rightarrow\{0,1\}\) which is certainly not surjective.
                        \item True by definition of what a function is.
                        \item False: \(\{(0,0)\}\) is a relation (and a function) and a
                             subset of \(\mathbb{R}\times\mathbb{R}\) but the domain
                            and range are merely \(\{0\}\subsetneq\mathbb{R}\).
                        \item True: if \((x,y)\in R\), then \(\{(x,y)\}\subseteq R\) is
                             a function contained in \(R\) from the set \(\{x\}\) to
                            the set \(\{y\}\).
                \end{enumerate}
        \end{solution}
\end{document}
```

This produces the following document. In my opinion, this document and its .tex file look nicer than it would have if we merely changed Figure 1 • 1 with these new problems, which is reproduced after.

My New Homework

A. Person

September 30, 2021

Problem 1 *Show that if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.*

Solution: Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah

---

Problem 3.2 *Show that the function $f : \mathbb{R} \to \mathbb{R}$ defined by*

$$f(x) = \begin{cases} x+1 & \text{if } x < 0 \\ 2x & \text{if } x \geq 0 \end{cases}$$

is not injective, but is surjective.

Solution: f is not injective because $f(-1/2) = 1/2 = f(1/4)$. To see that f is surjective, let $y \in \mathbb{R}$ be arbitrary. If $y \geq 0$, then $y/2 \geq 0$ and hence $x = y/2$ gives $f(x) = 2x = y$. If $y < 0$, then $y - 1 < 0$ so that $x = y - 1$ gives $f(x) = x + 1 = y$. Since $y \in \mathbb{R}$ was arbitrary, f is surjective.

Problem 3.4(a) *Which of the following is a true statement?*

1. *If $f : A \to B$ is a function, then f is always surjective onto its range.*
2. *If $f : A \to B$ is a function, then f is always surjective onto B.*
3. *If $f : A \to B$ is a function and $B \subseteq C$, then f is a function from A to C.*
4. *If $R \subseteq A \times B$ is a relation, then the domain of R is A and the range of R is B.*
5. *Every relation R has a subset $f \subseteq R$ such that f is a function.*

Solution:

1. True by definition of the range of f: the range of f is $\{y \in B : \exists x \in A\ (f(x) = y)\}$.
2. False: consider $f = \{(0,0)\}$ the function $f : \{0\} \to \{0,1\}$ which is certainly not surjective.
3. True by definition of what a function is.
4. False: $\{(0,0)\}$ is a relation (and a function) and a subset of $\mathbb{R} \times \mathbb{R}$ but the domain and range are merely $\{0\} \subsetneq \mathbb{R}$.
5. True: if $(x,y) \in R$, then $\{(x,y)\} \subseteq R$ is a function contained in R from the set $\{x\}$ to the set $\{y\}$.

1

My New Homework

A. Person

September 30, 2021

Problem 1: Show that if $A \subseteq B$ and $B \subseteq C$ then $A \subseteq C$.

Solution: Blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah blah

Problem 3.2: Show that the function $f : \mathbb{R} \to \mathbb{R}$ defined by

$$f(x) = \begin{cases} x+1 & \text{if } x < 0 \\ 2x & \text{if } x \geq 0 \end{cases}$$

is not injective, but is surjective.

Solution: f is not injective because $f(-1/2) = 1/2 = f(1/4)$. To see that f is surjective, let $y \in \mathbb{R}$ be arbitrary. If $y \geq 0$, then $y/2 \geq 0$ and hence $x = y/2$ gives $f(x) = 2x = y$. If $y < 0$, then $y - 1 < 0$ so that $x = y - 1$ gives $f(x) = x + 1 = y$. Since $y \in \mathbb{R}$ was arbitrary, f is surjective.

Problem 4: Which of the following is a true statement?

1. *If $f : A \to B$ is a function, then f is always surjective onto its range.*
2. *If $f : A \to B$ is a function, then f is always surjective onto B.*
3. *If $f : A \to B$ is a function and $B \subseteq C$, then f is a function from A to C.*
4. *If $R \subseteq A \times B$ is a relation, then the domain of R is A and the range of R is B.*
5. *Every relation R has a subset $f \subseteq R$ such that f is a function.*

Solution:

1

1. True by definition of the range of f: the range of f is $\{y \in B : \exists x \in A\ (f(x) = y)\}$.
2. False: consider $f = \{(0,0)\}$ the function $f : \{0\} \to \{0,1\}$ which is certainly not surjective.
3. True by definition of what a function is.
4. False: $\{(0,0)\}$ is a relation (and a function) and a subset of $\mathbb{R} \times \mathbb{R}$ but the domain and range are merely $\{0\} \subsetneq \mathbb{R}$.
5. True: if $(x,y) \in R$, then $\{(x,y)\} \subseteq R$ is a function contained in R from the set $\{x\}$ to the set $\{y\}$.

Now of course, all of this has been style preferences, and one might genuinely prefer the original approach. Regardless, defining environments and commands is a useful skill when working with LaTeX. Let us now examine that LaTeX code above, because it has introduced a fair amount of new things to consider.

- Some environments have inputs. Those inputs are placed directly after the environment begins, and (as with anything you wanted grouped together in TeX) enclosed in `{` and `}`.
- Aligning things in LaTeX, whether in a table, a matrix, or whatever, usually is written row by row with the separations between columns given by `&` and the separations between rows given by the command for a new line, `\\`. For example,

```
\(\begin{bmatrix}
        a & b\\
        c & d
\end{bmatrix}\)
```

  produces "$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$". This can also be seen in the `cases` environment above, which acts just the same:

```
\(\begin{cases}
        a & b\\
        c & d
\end{cases}\)
```

  produces "$\begin{cases} a & b \\ c & d \end{cases}$".
- The environment `enumerate` allows one to create numbered lists. The environment `itemize` allows one to create unnumbered lists like this one.
  - These lists can also be nested like this,
    - and this, which changes the bullet style and numbering style automatically.

  The general format is as follows.

```
\begin{enumerate}
        \item ...
        \item ...
        \item ...
\end{enumerate}
```

  If one wants to change the number, there is an *optional* argument that can follow the `\item`, e.g., adding the line `\item[(blah)] words words words` will output

  1. ...
  2. ...
  3. ...

  (blah) words words words

  In general, optional arguments are enclosed with brackets, `[` and `]`, rather than braces, `{` and `}`.

Section 3. Other Things You Might Want to Do

Here is a list of common things you might want to do, and how to do them.

- **Write a fraction** - use `\frac` which takes two arguments: `\(\frac{a}{b}\)` produces "$\frac{a}{b}$".
- **Start a new page** - use `\newpage` or `\pagebreak` (which will signal the computer to try and stretch the blank spaces to reach the end of the page similar to full justification as in this paragraph as opposed to just left justified with a ragged right edge to a paragraph).
- **Number equations** - use the `equation` environment which will automatically number equations. Alternatively, use a `\tag` command inside a display style equation, e.g. `\[A\in B \tag{a}\]` produces
$$A \in B \tag{a}$$
where we can also insert math into the tag, e.g. `\[A\in B\tag{\(*\)}\]` to have the tag be $(*)$ instead of (a). Also generally speaking, if an environment `blah` numbers equations, then the variant `blah*` (adding a *) doesn't number equations.
- **Change parantheses sizes** - use `\left` and `\right` to match parantheses sizes to what's contained in them. Alternatively, simply change the font size of the parantheses using the commands (from smallest to largest)
  - `\big`
  - `\Big`
  - `\bigg`
  - `\Bigg`

  For example, `(\frac{a}{b})`, `\left(\frac{a}{b}\right)`, and `\Bigg(\frac{a}{b}\Bigg)` produce (inside of `\[` and `\]`)
$$(\frac{a}{b}) \qquad \left(\frac{a}{b}\right) \qquad \Bigg(\frac{a}{b}\Bigg)$$
  respectively.
- **Write small math bigger** - if you write an integral or fraction in text with `\(` and `\)`, it produces something scaled to attempt to fit it on the line. To make it bigger as if you had used `\[` and `\]`, use the command `\displaystyle`. Compare `\frac{A}{B}` producing $\frac{A}{B}$ with `\displaystyle\frac{A}{B}` producing $\displaystyle\frac{A}{B}$. There are similar results for other big operators like summations, integrals, products, and so on: `\(\sum_{n=0}^\infty r^n\)` gives $\sum_{n=0}^{\infty} r^n$ compared with `\(\displaystyle\sum_{n=0}^\infty r^n\)` giving $\displaystyle\sum_{n=0}^{\infty} r^n$.
- **Write a math operator like sin or lim** - Several commands like this are baked into LaTeX. For example, with limits, `\(\lim_{n\rightarrow\infty}\)` produces $\lim_{n\rightarrow\infty}$ and in display style, $\displaystyle\lim_{n\rightarrow\infty}$. Among these are `\sin`, `\cos`, `\sup`, `\inf`, and many more. To create your own, use the package `amsmath` and use either
  1. `\DeclareMathOperator{\Command}{Written Text}` to define the command `\Command` that outputs "WrittenText", especially if you will use the command a lot, or
  2. `\operatorname{Written Text}` to output a single instance.
- **Include a space in math** - Use the command `\␣`, that is, a backslash followed by a space. There are other spacing commands too, including `\!` which gives a *negative* space.
- **Align multiple equations** - use the `align*` environment (or `align` if you want the equations numbered) with the same syntax as with a table or matrix:

```
\begin{align*}
        f(x)&=x+1+1\\
        &=x+2
\end{align*}
```

produces

$$f(x) = x + 1 + 1$$
$$= x + 2$$

Essentially, things are lined up on the `&` symbol. One can get more complicated with multiple columns, each of which is separated by another `&` symbol. For example,

```
\begin{align*}
        f(x)&=x+1+1     &       g(x)&=x-1-1\\
        &=x+2   &       &=x-2
\end{align*}
```

produces

$$f(x) = x + 1 + 1 \qquad g(x) = x - 1 - 1$$
$$= x + 2 \qquad = x - 2$$

- **Have multiple display equations without the extra space** - You might write multiple display style expressions that then have excess space between them. Use the `gather*` (or `gather` if you want things numbered) to collect the equations together without the extra space.

```
\begin{gather*}
        A=\{x \\:x^2-1=0\}\]\\
        B=\{c \\:c ^2\A \\\}
\end{gather*}
```

produces

$$A = \{x \in \mathbb{R} : x^2 - 1 = 0\}$$
$$B = \{c \in \mathbb{R} : c^2 \in A \setminus \mathbb{R}\}$$

- **Change paper size** - use the package `geometry` and look up the parameters you'd like on the internet or the package documentation.
- **Change fonts** - This is complicated. For basic stuff, `\textt` writes things in a `typewriter font`, and `\textsf` writes things in a san serif font. There are various fonts given by the `amsfonts` package, like fraktur $\mathfrak{A}$, and script $\mathcal{A}$. In general, use a package that can do it for you, or else use XeLaTeX (XeLaTeX) with the package `fontspec`.
- **Write greek letters** - use the command with the same name as the greek letter: `\(\pi\)` produces π. Capitalize the command to produce the capital letter: `\(\Gamma\)` produces Γ whereas `\(\gamma\)` produces γ. Several greek letters also have variant forms like ϕ and φ, which can be produced with a `var` in the command: `\varphi` gives φ, `\varepsilon` gives ε instead of ϵ, `\varbeta` gives $\varbeta$ instead of β, and so on. The capital letters with the same form as their Roman equivalents have no special command: there is no command `\Alpha`. Instead, one can simply write "A" in an upright form.
- **Write math letters in an upright or bold form** - To write upright letters, use `\mathrm` ("rm" for "roman"). To write bold letters, use `\mathbf` or `\boldsymbol` or `\bm` with the package `boldmath`. The issue is that "bold" versions require a separate font that most font creators have not bothered with. So at best, much of the time a bold symbol is just an approximation to a proper bold version.
- **Insert graphics or pictures** - use the package `graphicx` and use the command `\insertgraphics{...}` where you place in the filename of the picture or pdf you want inserted. Note that often this will not be scaled correctly, which you can change with an optional argument: `\insertgraphics[scale=.75]{file.png}` outputs the picture at 75% scale. Note also that usually you want to center this inserted graphic.
- **Center things** - use the environment `center` or the command `\centerline`.
- **Add hyperlinks and referencing** - use the package `hyperref` which is very complicated and is probably better explained elsehwere on the internet.
- **Draw diagrams** - use the package `tikz` and the environment `tikzpicture`. This is often very tedious but the result looks very nice and is more precise than a raster image inserted into the document. Again, elsewhere on the internet will have better tutorials on how to use Ti*k*Z than I can give here.
- **Write a special character like '\'** - There are a lot of symbols that LaTeX reserves for other purposes:
  - ~ (`\textasciitilde`) is used to create a space that a new line is now allowed to break on.

  - \ (\textbackslash) is used to mark the start of a command.
  - _ (\textunderscore) is used in math to create subscripts.
  - ^ (\textasciicircum) is used in math to create superscripts.
  - $ (\$) is used in TEX to enter and exit math mode.
  - % (\%) is used to mark that the rest of the line is a comment.
  - # (\#) is used to denote the argument in the definition of a new command or environment.
  - & (\&) is used to separate columns in a table, matrix, array, or what have you.
- **Write code** - use the environment verbatim or the in-line version with the command verb. Note that the syntax of verb is different from most commands: you can enclose the argument in just about every symbol except '*', this allows you to write { with ease. Most use a vertical line as the delimiter: \verb|...|, but there isn't anything truly special about this. For a version that marks spaces, use a starred-variant: \verb*|a and b| produces a␣and␣b. For more in-depth writing of code, use the package listings.
- **Add space between paragraphs** - Add \setlength{parskip}{10pt} to the preamble of the document where the 10pt space was arbitrary and can be added to anything. This can also be added to the itemize and enumerate environments to add space between the items and have paragraphs within items.
- **Write out dashes** - there are three lengths to a dash commonly: a hypen -, an en dash – (for ranges of numbers like "pages 3–40"), an em dash — (for parenthetical-like phrases). These can be typed by chaining hyphens together: - produces a hyphen, -- produces an en dash, and --- produces an em dash.
- **Create a footnote** - use the command footnote.
- **Write accents** - in text, accents can be written using commands like \'A or \^A to produce 'Á' and 'Â' respectively. The accents in math mode are often distinct. The common accents include the following.

Accent	Text mode	Math Mode Command	Example
grave	\`	\grave	ś, $\grave{s}$
acute	\'	\acute	ś, $\acute{s}$
double acute	\H	doesn't exist	s̋
dot	\.	\dot	ṡ, $\dot{s}$
dieresis	\"	\ddot	s̈, $\ddot{s}$
circumflex	\^	\hat	ŝ, $\hat{s}$
tilde	\~	\tilde	s̃, $\tilde{s}$
breve	\u	\breve	s̆, $\breve{s}$
check	\v	\check	š, $\check{s}$
cedilla	\c	doesn't exist	ş
ogonek	\k	doesn't exist	ą
dot under	\d	doesn't exist	ṣ
macron	\=	\bar	s̄, $\bar{s}$
arrow above	doesn't exist	\vec	$\vec{s}$

3 • 1. Table 1: Common Accents in TEX

Now I want to turn my attention to a few minor style issues that come up. These aren't at all big deals, but they do highlight some deeper parts of how TEX works, and can be annoying to the people that notice them and can't figure out how to get them to go away.

Several common issues include modifying spacing. In particular, if you have a period followed by a space, TEX automatically assumes you're starting a new sentence. So an abbreviation like Dr. Jones will produce "Dr. Jones" instead of the appropriately spaced "Dr. Jones" which is created with the commend \␣: Dr.\ Jones. There are other space commands like \. and \, and so on which create spaces of various lengths.

A related concern about spacing is with commands that have no inputs. In some sense, all commands[ii] require a character to tell TEX when the name for the command has ended. What's really going on under the hood is that inputs

[ii] With names that aren't special characters like \(

are taken in character by character. For example, one could write `\(\frac{1}{2}\)` for $\frac{1}{2}$, but one could also just write `\(\frac12\)`. The parantheses group together things that should be taken as a single input. For example, `\(\frac{1+1}{2}\)`, giving $\frac{1+1}{2}$, cannot be written `\(\frac1+12\)`. On the otherhand, `\(\fracab\)` doesn't produce $\frac{a}{b}$ like `\(\frac12\)` does with $\frac{1}{2}$. One must write `\(\frac{a}{b}\)` instead. So what gives? Parantheses tell TEX when you have finished writing a command, but special characters like numbers, and *spaces* also do this. So with a command like `\TeX`, which produces "TEX", one must be careful about ensuring the space that follows it isn't eaten up as if TEX is interpretting it merely as the end of the command rather than an actual space. For example, compare `\TeX␣is␣cool`, producing "TEXis cool", with `\TeX{}␣is␣cool` or `\TeX\␣is␣cool`, producing "TEX is cool".

Another issue is with quotation marks: `'something'` produces 'something', with the same quotation mark twice. Instead, one should type `` ` `` for the other quotation mark: `` `something' `` produces 'something'. These can also be doubled up to produce "quotes like this".