This document shows the ingredients of a good assignment or project report for this (part of the) course. The \LaTeX source code of this document exemplifies almost everything you need to know about \LaTeX in order to typeset a professional-looking report (for this course). Use it as a starting point for imitation and delete everything irrelevant. The usage of \LaTeX is\textit{optional}, but highly recommended, for reasons that will soon become clear to those who have never used it before: any learning time is\textit{outside} the time budget of this course, but will hugely pay off, if not in this course then in the next courses you take and when writing a thesis or other report.

Address each task of each problem, using the numbering and ordering in which the problems and tasks appear in the assignment statement:

- For each task requiring a model, follow the advice and structure of Section C.
- For each task requiring an evaluation, follow the structure of Section D.

Delete all unnecessary text to save trees and suitably replace all other text in this document.

Part 1

The Minimal Magic Square Problem

All experiments were run under Linux Ubuntu 18.04 (64 bit) on an Intel Xeon E5520 of 2.27 GHz, with 4 processors of 4 cores each, with a 70 GB RAM and an 8 MB L3 cache (a ThinLinc computer of the IT department).\textsuperscript{1} (If different hardware was used for different tasks, then justify this and replicate a paragraph like this one within each relevant part or section.)

A Heading for Task A

(Write your answer for Task A here.)

\textsuperscript{1}Hint: Under Linux, do lscpu to find this information. Under macOS, you find this information via “About This Mac” in the Apple menu.
B Use Appropriate Headings for the Sections

(Remember that this document is a guideline, so you should change its text when appropriate.)

C Model

(You need not describe a problem specified in an assignment statement, but you must describe the problem of your project.) The Minimal Magic Square problem is about finding a magic square (as defined in Topic 5), of a given order, whose sum of the four corner elements is minimal.

Our model, with the prescribed comments, is uploaded as file minMagicSquare.mzn and given in Listing 1. A MiniZinc [1] model must include comments that explain:

- the parameters that are not part of the possibly provided skeleton code;
- the decision variables, and which problem constraints, if any, they automatically enforce;
- the redundant decision variables (specify: mutually or not?); if none, justify this below;
- the problem constraints not enforced by the choice of decision variables;
- the objective, including possibly the objective function;
- the channelling constraints (specify: one-way or two-way?); if none, then justify this below;
- the implied constraints; if none, then justify this below;
- the symmetry-breaking constraints; if none, then justify this below;
- the CP/LCG inference annotations; if none, then justify this below;
- the CP/LCG search annotation; if none, then justify this below.

The quality of model comments is considered while grading. You must include your model in your report in addition to uploading it as a separate file (so that we can run it).

Listing 1: A MiniZinc model for the Minimal Magic Square problem

```
int: n; % the width (and height) of the magic square
set of int: N = 1..n; % the index range for the rows and columns

% Magic[r,c] = the value at row r and column c of the magic square:
array[N,N] of var 1..n*n: Magic;

% All values in the magic square are different:
constraint alldifferent(Magic) :: domain;

% The sum of each row is equal to the magic sum:
constraint forall(r in N)(sum(Magic[r,..]) = magicSum);

% The sum of each column is equal to the magic sum:
constraint forall(c in N)(sum(Magic[..,c]) = magicSum);

% The sum of each major diagonal is equal to the magic sum:
constraint sum([Magic[i,i] | i in N]) = magicSum;
constraint sum([Magic[n+1-i,i] | i in N]) = magicSum;
```
% Each row sums up to the magic sum, there are n rows,
% and their sum is equal to the sum of the entire magic square,
% hence the following implied constraint fixes magicSum
% and is useful under all solving technologies:
constraint n * magicSum = sum(1..n*n); % implied constraint

% Break the three rotation symmetries and three of the four
% reflection symmetries by requiring the top-left corner to
% be smaller than the other corners:
constraint symmetry_breaking_constraint(
    Magic[1,1] < Magic[1,n] /
    Magic[1,1] < Magic[n,1] /
    Magic[1,1] < Magic[n,n]
);

% Break the down-right-diagonal reflection symmetry by requiring the
% bottom-left corner to be smaller than the top-right corner:
constraint symmetry_breaking_constraint(Magic[n,1] < Magic[1,n]);

solve
% Search first on 4 corners, subject to objective function (below)
% and more constraints (diagonals + symmetry) than the other cells,
% in lower halves of bisections, due to arithmetic and minimisation:
:: seq_search(
    int_search([Magic[i,j] | i,j in {1,n}],
        input_order,indomain_split),
    int_search(Magic,occurrence,indomain_split))
% Minimise the sum of the 4 corners of the magic square:
minimize sum([Magic[i,j] | i,j in {1,n}]);

output
["Magic sum: ", show(magicSum), "\nMagic square: 
"] ++
[show(Magic[r,c]) ++ if c=n then "\n" else " " endif | r,c in N]++
["CornerSum: ", show(sum([Magic[i,j] | i,j in {1,n}]))], "\n"]
++ ["Corners: ", show([Magic[i,j] | i,j in {1,n}])];

(Provide, here in the report, a more detailed explanation (i) why each redundant decision variable is deemed useful, (ii) why each channelling, implied, and symmetry-breaking constraint is correct, as well as (iii) how the inference and search annotations were chosen. Use any combination of in-lined code, mathematical notation, and plain English to explain your choices.)

Redundant Decision Variables and Channelling Constraints. (A valid excuse for omitting redundant decision variables and their channelling constraints is that Topic 4 has not been covered yet.) We were unable to define any (mutually or non-mutually) redundant decision variables where experiments (not reported here) revealed that these variables and appropriate (one-way or two-way) channelling constraints accelerate at least one backend.

Implied Constraints. (A valid excuse for omitting implied constraints is that Topic 4 has not been covered yet.) Since the sum of each row must be equal to the magic sum, and since
all values from 1 to $n^2$ must occur exactly once in the magic square, it is implied that the sum of the rows is the sum of the entire magic square:

```
constraint n * magicSum = sum(1..n*n); % implied constraint
```

Experiments (not reported here) revealed that this implied constraint accelerates all backends of all the considered solving technologies, as it actually fixes the decision variable `magicSum` since $n$ is a parameter, and hence we do not flag it using the recommended `implied_constraint` special predicate.

**Symmetry-Breaking Constraints.** (A valid excuse for omitting symmetry-breaking constraints, except for Task C of Problem 6 of Assignment 1, is that Topic 5 has not been covered yet.) A magic square has rotation and reflection symmetries. The rotation symmetries for 90, 180, and 270 degrees and the reflection symmetries for the horizontal axis, vertical axis, and up-right diagonal can be broken by requiring the top-left corner of the magic square to be smaller than the other corners:

```
constraint symmetry_breaking_constraint(
    Magic[1,1] < Magic[1,n] /
    Magic[1,1] < Magic[n,1] /
    Magic[1,1] < Magic[n,n]
);
```

The reflection symmetry for the down-right diagonal can be broken by requiring the bottom-left corner to be smaller than the top-right corner:

```
constraint symmetry_breaking_constraint(Magic[n,1] < Magic[1,n]);
```

As recommended, we flag these constraints using the `symmetry_breaking_constraint` special predicate. Experiments (not reported here) revealed that these symmetry-breaking constraints enormously accelerate backends that enable them.

**Inference Annotations.** (A valid excuse for omitting inference annotations is that Topic 8 has not been covered yet.) We leave the sensible choice of inference for the linear equality constraints (lines 14 to 19) and the linear objective function (line 48) to each CP and LCG backend, assuming it is domain consistency for smaller values of parameter $n$ and bounds consistency for its larger values, the threshold being probably backend-specific.

The implied linear equality constraint (line 25) has only one decision variable and functionally determines it, hence this constraint will be immediately satisfied and needs no annotation.

All the symmetry-breaking constraints (lines 30 to 37) are inequalities on only two decision variables, where we assume domain consistency under each CP and LCG backend, as this is as cheap to achieve as the weaker bounds consistency.

For the all-difference constraint (line 11) we explicitly suggest domain consistency, as experiments (not reported here) revealed that this accelerates all CP and LCG backends.

**Search Annotation.** (A valid excuse for omitting a search annotation is that Topic 8 has not been covered yet.) Since the four corners of the magic square correspond to the only decision variables of the objective function and even occur in more constraints than the other cells of the magic square, namely additionally in the diagonal and symmetry-breaking constraints, we use a sequence of two search phases (lines 43 to 46), based on experiments (not reported here):

1. We search on the four corners. We use the variable selection strategy `input_order` as the order of those four decision variables does not matter. We use the value selection
strategy `indomain_split` in order to bisect the domain of the chosen decision variable, because bisection is known to be good in the presence of arithmetic constraints (only the all-difference constraint in line 11 is not arithmetic), starting on the lower half, because the objective function is to be minimised.

2. We search on the remaining variables. We use the strategy `occurrence` to select a decision variable occurring in the largest number of constraints, as the non-corners of the diagonals are more constrained than the other remaining cells. We use the value selection strategy `indomain_split` for the same reason as in the first phase.

### D Evaluation

(Give a table of experiment results, in exactly the given format, for the chosen backends, and analyse it.) Table 1 gives the results for various values of parameter $n$ in our model.\(^2\) The time-out was 600,000 milliseconds. (Which solvers win overall, and how do you draw that conclusion?) We observe that Gurobi wins overall, as it is the only one not to time out for the largest chosen instance, with $n = 6$. (Which solvers scale best, and how do you draw that conclusion?) Gurobi and Picat scale best, as they are the only ones to establish feasibility for $n = 6$, even though the Picat objective value of 39 at time-out is far above the minimum of 26 proven by Gurobi. (How do the solvers scale, and how do you draw that conclusion?) On the small instances, with $n \leq 4$, Gecode wins, narrowly defeating Chuffed followed by Gurobi and Picat. Starting from the medium instance, with $n \geq 5$, Gurobi clearly wins, with only Gecode also not timing out for $n = 5$, with Chuffed finding but not proving the minimum for $n = 5$, and with Picat missing the minimum already by one unit for $n = 5$. (Does the difficulty of instances monotonically increase with their size, and how do you draw that conclusion?) Picat is the only solver where $n = 4$ is harder than the smaller $n = 3$. Observe in the Gurobi column that the minimum objective value also does not monotonically increase with $n$. (How suitable is local search compared to systematic search, and how do you draw that conclusion?) The backend fzn-oscar-cbld always times out, because local search can by construction not prove minima on problems, such as here, where the trivial lower bound (namely $10 = 1 + 2 + 3 + 4$ here) on the objective value is not feasible; it finds the known minima for $n \leq 4$, but is far above the known minimum for $n = 5$ and cannot even establish feasibility for $n = 6$ before timing out.

---

\(^2\)You must use the script explained in the current cheatsheet at http://user.it.uu.se/~gusbj192/courses/M4CO: it conducts the experiments and generates a result table (see the \LaTeX\ source code of Table 1) that is automatically imported (rather than manually copied) into your report: each time you change the model, it suffices to re-run that script and re-compile your report, without any tedious number copying!
Table 1: Results for our Minimal Magic Square model. In the ‘time’ column, if the reported time is less than the time-out (600,000 milliseconds here), then the reported objective value in the ‘obj’ column was proven optimal; else the time-out is indicated by ‘t/o’ and the reported objective value is either the best value found, but not proven optimal, before timing out, or ‘–’, indicating that no feasible solution was found before timing out. (Only include the following sentence if experiments are run over a range (not a set) of instances:) If the reported time is ‘–’, then that instance was not run on that backend, as the latter timed out on a smaller instance. (This is normally not a correct assumption, as larger instances can be easier, but this assumption is fine for the assignments of this course.) Achieve the following by manual editing: Boldface indicates the best performance on each row.

<table>
<thead>
<tr>
<th>Backend</th>
<th>Gecode n</th>
<th>obj</th>
<th>time</th>
<th>Chuffed</th>
<th>obj</th>
<th>time</th>
<th>Gurobi</th>
<th>obj</th>
<th>time</th>
<th>fzn-oscar-cbls</th>
<th>obj</th>
<th>time</th>
<th>Picat</th>
<th>obj</th>
<th>time</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td></td>
<td>20</td>
<td>422</td>
<td>20</td>
<td>954</td>
<td></td>
<td>20</td>
<td>1268</td>
<td></td>
<td>20</td>
<td>t/o</td>
<td></td>
<td>20</td>
<td>1262</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>34</td>
<td>372</td>
<td>34</td>
<td>680</td>
<td></td>
<td>34</td>
<td>1210</td>
<td></td>
<td>34</td>
<td>t/o</td>
<td></td>
<td>34</td>
<td>8297</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>26</td>
<td>68100</td>
<td>26</td>
<td>t/o</td>
<td></td>
<td>26</td>
<td>46645</td>
<td>36</td>
<td>t/o</td>
<td>27</td>
<td>t/o</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>t/o</td>
<td></td>
<td></td>
<td></td>
<td>26</td>
<td>65681</td>
<td></td>
<td>t/o</td>
<td>39</td>
<td>t/o</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Results for our Minimal Magic Square model. In the ‘time’ column, if the reported time is less than the time-out (600,000 milliseconds here), then the reported objective value in the ‘obj’ column was proven optimal; else the time-out is indicated by ‘t/o’ and the reported objective value is either the best value found, but not proven optimal, before timing out, or ‘–’, indicating that no feasible solution was found before timing out. (Only include the following sentence if experiments are run over a range (not a set) of instances:) If the reported time is ‘–’, then that instance was not run on that backend, as the latter timed out on a smaller instance. (This is normally not a correct assumption, as larger instances can be easier, but this assumption is fine for the assignments of this course.) Achieve the following by manual editing: Boldface indicates the best performance on each row.
Feedback to the Teachers

(Please write a paragraph, which will not be graded, describing your experience with this assignment or project: which aspects were too difficult or too easy, and which aspects were interesting or boring? This may help us improve the course for the next year.)

Error Report

(Your model must compile and run error-free under backends of all the considered solving technologies, unless you have the teaching assistant’s prior approval to upload an error report here. For example:)

largecumulative.mzn  For the larger instances, mzn-oscar-cbls crashes with the following exception, which seems to be caused by the JVM not being allocated enough heap space:

```
>./fzn-oscar-cbls -s -t 600 /tmp/tmp.fzn
Exception in thread "main" java.lang.OutOfMemoryError: Java heap space
at scala.collection.mutable.FlatHashTable$class.growTable(FlatHashTable.scala:217)
at scala.collection.mutable.FlatHashTable$CBSclass.addEntry(FlatHashTable.scala:159)
at scala.collection.mutable.FlatHashTable$CBSclass.addEntry(FlatHashTable.scala:140)
at scala.collection.mutable.FlatHashTable$CBSclass.addEntry(FlatHashTable.scala:139)
at scala.collection.mutable.HashSet.addElem(HashSet.scala:40)
at scala.collection.mutable.HashSet.addElem(HashSet.scala:59)
at oscar.cp.scheduling.constraints.EnergeticReasoning$3.addNonfun$computeIntervals$11$nonfun$apply$ImcvI3sp2.apply(EnergeticReasoning.scala:127)
at oscar.cp.scheduling.constraints.EnergeticReasoning$3.addNonfun$computeIntervals$11$nonfun$apply$ImcvI3sp2.apply(EnergeticReasoning.scala:126)
at oscar.cp.scheduling.constraints.EnergeticReasoning$3.addNonfun$computeIntervals$11$nonfun$apply$ImcvI3sp2.apply(EnergeticReasoning.scala:126)
at scala.collection.TraversableLike$WithFilter$$anonfun$foreach$1.apply(TraversableLike.scala:778)
at scala.collection.mutable.HashSet.foreach(HashSet.scala:78)
at oscar.cp.scheduling.constraints.EnergeticReasoning.computeIntervals(EnergeticReasoning.scala:126)
at oscar.cp.scheduling.constraints.EnergeticReasoning.propagate(EnergeticReasoning.scala:77)
```

References

Checklist before Submitting

In order to protect yourself against an unnecessary loss of points, use the following checklist before submitting:

- Crosscheck your models against the checklists at the ends of the slides of Topics 2 and 3.
- Crosscheck your report against the assignment instructions.
- Remember that when submitting you implicitly certify (a) that your report and all its uploaded attachments were produced solely by your team, except where explicitly stated otherwise and clearly referenced, (b) that each teammate can individually explain any part starting from the moment of submitting your report, and (c) that your report and attachments are not freely accessible on a public repository.
- Spellcheck all documents, including the comments in the source code.
- Proofread, if not grammar-check, your report at least once per teammate.

More \LaTeX and Technical Writing Advice

Unnumbered itemisation (only to be used when the order of the items does not matter):³

- Unnumbered displayed formula:
  \[ E = m \cdot c^2 \]

- Numbered displayed formula, which is cross-referenced somewhere:
  \[ E = m \cdot c^2 \]

- Formula — the same as formula (D) — spanning more than one line:
  \[
  E = m \cdot c^2
  \]

Numbered itemisation (only to be used when the order of the items does matter):

1. First do this.
2. Then do that.
3. If we are not finished, then go back to Step 2, else stop.
Figure 1: A binary search tree (on the left), a binary min-heap (in the middle), and a binomial tree of rank 3 (on the right).

```
1: function $f(n)$
2: if $n < 0$ then // optional comment
3:   $n := -2 \cdot n$ // optional comment
4: else // $n \geq 0$
5:   $n := 3 \cdot n$
6: while $n > 0$ do // optional comment
7:   $n := n - 1$
8: return $n$
```

Algorithm 1: Silly algorithm

Tables and elementary mathematics are typeset as exemplified in Table 2; see ftp://ftp.ams.org/pub/tex/doc/amsmath/short-math-guide.pdf for many more details.

Use \textit{...} in mathematical mode for each multiple-letter identifier in order to avoid typesetting the identifier like the product of single-letter ones. For example, note the typographic difference between the identifier $WL$, obtained through $\textit{WL}$, and the product $WL$, where there is a small space between the $W$ and the $L$, obtained through $WL$.

Do not use programming-language-style lower-ASCII notation (such as ! for negation, && for conjunction, || for disjunction, and the equality sign = for assignment) in algorithms or formulas (but rather use \texttt{not}, \texttt{and}, \texttt{or}, and \texttt{←} or $\left\lvert\cdot\right\rvert$, respectively), as this testifies to a very strong confusion of concepts.

Figures can be imported with \includegraphics or drawn inside the \LaTeX source code using the highly declarative notation of the tikz package: see Figure 1 for sample drawings. It is perfectly acceptable in this course to include scans or photos of drawings that were carefully done by hand.

Algorithms can be typeset as pseudo-code as exemplified in Algorithm 1: study its \LaTeX source code.

If you are not sure whether you will stick to your current choice of notation or terminology, then introduce a new (possibly parametric) command. For example, upon

\texttt{\newcommand{\Cardinality}[1]{\left\lvert#1\right\rvert}}

Use footnotes very sparingly, and note that footnote pointers are never preceded by a space and always glued immediately behind the punctuation, if there is any.
the formula $\text{Cardinality}(S)$ typesets the cardinality of set $S$ as $|S|$ with autosized vertical bars and proper spacing, but upon changing the definition of that parametric command to

\begin{verbatim}
\newcommand{\Cardinality}[1]{\# #1}
\end{verbatim}

and recompiling, the formula $\text{Cardinality}(S)$ typesets the cardinality of set $S$ as $\#S$. Similarly, upon

\begin{verbatim}
\newcommand{\MiniZinc}{\textit{Mini-Zinc}}
\end{verbatim}

the text \MiniZinc\ typesets into MiniZinc, hyphenation being only possible in the middle, but upon changing the definition of that non-parametric command to

\begin{verbatim}
\newcommand{\MiniZinc}{\textsc{Mini-Zinc}}
\end{verbatim}

and recompiling, the text \MiniZinc\ typesets into MINIZINC. You can thus obtain an arbitrary number of changes in the document with a constant-time change in its source code, rather than having to perform a linear-time find-and-replace operation within the source code, which is painstaking and error-prone. The imported file macros.tex has a lot of useful predefined commands about mathematics, CP, Gecode, modelling, MiniZinc, and algorithms.

Use commands on positioning (such as \hspace, \vspace, and \noindent) and appearance (such as \small for reducing the font size, and \textit for italics) very sparingly, and ideally only in (parametric) commands, as the very idea of mark-up languages such as \LaTeX\ is to let the class designer (usually a trained professional typesetter) decide on where things appear and how they look. For example, \textbf{emph} (for emphasis) compiles (outside italicised environments, such as theorem) into italics under the article class used for this document, but it may compile into \textbf{boldface} under some other class.

If you do not (need to) worry about how things look, then you can fully focus on what you are trying to express!

Note that no absolute numbers are used in the \LaTeX source code for any of the references inside this document. For ease of maintenance, \label is used for giving a label to something that is automatically numbered (such as an algorithm, equation, figure, footnote, item, line, part, section, subsection, or table), and \ref is used for referring to a label. An item in the bibliography file is referred to by \cite instead. Upon changing the text, it suffices to recompile, once or twice, and possibly to run BibTeX again, in order to update all references consistently.

Always write Table~\ref{tab:maths} instead of Table \ref{tab:maths}, by using the non-breaking space (which is typeset as the tilde $\sim$) instead of the normal space, because this avoids that a cross-reference is spread across a line break, as for example in “Table 2”, which is considered poor typesetting.

The rules of English for how many spaces to use before and after various symbols are given in Table 3. Beware that they may be very different from the rules in your native language.

\footnote{Feel free to report to the head teacher any other features that you would have liked to see discussed and exemplified in this template document.}
<table>
<thead>
<tr>
<th>Topic</th>
<th>\LaTeX{} code</th>
<th>Appearance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greek letter</td>
<td>$\Theta, \Omega, \epsilon$</td>
<td>$\Theta, \Omega, \epsilon$</td>
</tr>
<tr>
<td>multiplication</td>
<td>$m \cdot n$</td>
<td>$m \cdot n$</td>
</tr>
<tr>
<td>division</td>
<td>$\frac{m}{n}, m \div n$</td>
<td>$m \div n$</td>
</tr>
<tr>
<td>rounding down</td>
<td>$\left\lfloor n \right\rfloor$</td>
<td>$\lfloor n \rfloor$</td>
</tr>
<tr>
<td>rounding up</td>
<td>$\left\lceil n \right\rceil$</td>
<td>$\lceil n \rceil$</td>
</tr>
<tr>
<td>binary modulus</td>
<td>$m \bmod n$</td>
<td>$m \mod n$</td>
</tr>
<tr>
<td>unary modulus</td>
<td>$m \equiv n \mod \ell$</td>
<td>$m \equiv n \mod \ell$</td>
</tr>
<tr>
<td>root</td>
<td>$\sqrt{n}, \sqrt[k]{n}$</td>
<td>$\sqrt{n}, \sqrt[k]{n}$</td>
</tr>
<tr>
<td>exponentiation, superscript</td>
<td>$n^i$</td>
<td>$n^i$</td>
</tr>
<tr>
<td>subscript</td>
<td>$n_i$</td>
<td>$n_i$</td>
</tr>
<tr>
<td>overline</td>
<td>$\overline{n}$</td>
<td>$\overline{n}$</td>
</tr>
<tr>
<td>base 2 logarithm</td>
<td>$\log_2 n$</td>
<td>$\log_2 n$</td>
</tr>
<tr>
<td>base b logarithm</td>
<td>$\log_b n$</td>
<td>$\log_b n$</td>
</tr>
<tr>
<td>binomial</td>
<td>$\binom{n}{k}$</td>
<td>$\binom{n}{k}$</td>
</tr>
<tr>
<td>sum</td>
<td>$\sum_{i=1}^n i$</td>
<td>$\sum_{i=1}^n i$</td>
</tr>
<tr>
<td>numeric comparison</td>
<td>$\leq, &lt;, =, \neq, &gt;, \geq$</td>
<td>$\leq, &lt;, =, \neq, &gt;, \geq$</td>
</tr>
<tr>
<td>non-numeric comparison</td>
<td>$\prec, \nprec, \preceq, \succeq$</td>
<td>$\prec, \nprec, \preceq, \succeq$</td>
</tr>
<tr>
<td>extremum</td>
<td>$\min, \max, +\infty, \bot, \top$</td>
<td>$\min, \max, +\infty, \bot, \top$</td>
</tr>
<tr>
<td>function, tuple</td>
<td>$f : A \to B, \circ, \mapsto$</td>
<td>$f : A \to B, \circ, \mapsto$</td>
</tr>
<tr>
<td>sequence, tuple</td>
<td>$\langle a, b, c \rangle$</td>
<td>$\langle a, b, c \rangle$</td>
</tr>
<tr>
<td>set</td>
<td>${a,b,c}, \emptyset, \mathbb{N}$</td>
<td>${a,b,c}, \emptyset, \mathbb{N}$</td>
</tr>
<tr>
<td>set membership</td>
<td>$\in, \notin \in$</td>
<td>$\in, \notin \in$</td>
</tr>
<tr>
<td>set comprehension</td>
<td>${i \mid 1 \leq i \leq n}$</td>
<td>${i \mid 1 \leq i \leq n}$</td>
</tr>
<tr>
<td>set operation</td>
<td>$\cup, \cap, \setminus, \times$</td>
<td>$\cup, \cap, \setminus, \times$</td>
</tr>
<tr>
<td>set comparison</td>
<td>$\subset, \subseteq, \not\supset$</td>
<td>$\subset, \subseteq, \not\supset$</td>
</tr>
<tr>
<td>logic quantifier</td>
<td>$\forall, \exists, \nexists$</td>
<td>$\forall, \exists, \nexists$</td>
</tr>
<tr>
<td>logic connective</td>
<td>$\land, \lor, \neg, \Rightarrow$</td>
<td>$\land, \lor, \neg, \Rightarrow$</td>
</tr>
<tr>
<td>logic</td>
<td>$\models, \equiv, \vdash$</td>
<td>$\models, \equiv, \vdash$</td>
</tr>
<tr>
<td>miscellaneous</td>
<td>$&amp;$, $#$, \approx, \sim, \ell$</td>
<td>$&amp;$, $#$, \approx, \sim, \ell$</td>
</tr>
<tr>
<td>dots</td>
<td>$\ldots, \cdots, \vdots, \ddots$</td>
<td>$\ldots, \cdots, \vdots, \ddots$</td>
</tr>
<tr>
<td>dots (context-sensitive)</td>
<td>$1, \ldots, n; 1 + \ldots + n$</td>
<td>$1, \ldots, n; 1 + \ldots + n$</td>
</tr>
<tr>
<td>parentheses (autosizing)</td>
<td>$\left(m\cdot(n\cdot k)\right), (m\cdot(n\cdot k))$</td>
<td>$(m^n)^k, (m^n)^k$</td>
</tr>
<tr>
<td>identifier of &gt; 1 character</td>
<td>$\text{identifier}$</td>
<td>$\text{identifier}$</td>
</tr>
<tr>
<td>hyphen, n-dash, m-dash, minus</td>
<td>$-, --, ---, -$</td>
<td>$-, --, ---, -$</td>
</tr>
</tbody>
</table>

Table 2: The typesetting of elementary mathematics. Note very carefully when italics are used by \LaTeX{} and when not, as well as all the horizontal and vertical spacing performed by \LaTeX{}.

<table>
<thead>
<tr>
<th>number of spaces after</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>number of spaces before</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3: Spacing rules of English