

L^AT_EX for linguists

Sylvia Blaho & Patrik Bye

The logo for the Center for Advanced Study in Theoretical Linguistics (CASTL) features the letters C, A, S, T, and L in a white, serif font, spaced out across a dark green rectangular background.

CASTL

Center for Advanced Study
in Theoretical Linguistics

University of Tromsø

Spring 2004

Class 5: Fine-tuning text & tables

- Fine-tuning text

Class 5: Fine-tuning text & tables

- Fine-tuning text
 - Word and character spacing

Class 5: Fine-tuning text & tables

- Fine-tuning text
 - Word and character spacing
 - Vertical spacing

Class 5: Fine-tuning text & tables

- Fine-tuning text
 - Word and character spacing
 - Vertical spacing
 - Line breaking

Class 5: Fine-tuning text & tables

- Fine-tuning text
 - Word and character spacing
 - Vertical spacing
 - Line breaking
 - Page breaking

Class 5: Fine-tuning text & tables

- Fine-tuning text
 - Word and character spacing
 - Vertical spacing
 - Line breaking
 - Page breaking
- Tables

1. Fine-tuning text

1.1. Word and character spacing

- (1) $\text{\TeX}$ inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.

1. Fine-tuning text

1.1. Word and character spacing

- (1) T_EX inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.
- (2) T_EX interprets a period following a lowercase letter as the end of a sentence, **Prof. Svenonius** should be typeset as **Prof.**_Svenonius.

1. Fine-tuning text

1.1. Word and character spacing

- (1) T_EX inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.
- (2) T_EX interprets a period following a lowercase letter as the end of a sentence, **Prof. Svenonius** should be typeset as **Prof.**_Svenonius.
- (3) An uppercase letter followed by a period is interpreted as not being at the end of a sentence, so _ has to be inserted before the period to ensure extra spacing: **at CASTL.** is to be typeset as **at_CASTL_.**

1. Fine-tuning text

1.1. Word and character spacing

- (1) T_EX inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.
- (2) T_EX interprets a period following a lowercase letter as the end of a sentence, **Prof. Svenonius** should be typeset as `Prof.\_Svenonius`.
- (3) An uppercase letter followed by a period is interpreted as not being at the end of a sentence, so `\@` has to be inserted before the period to ensure extra spacing: **at CASTL.** is to be typeset as `at\_CASTL\@..`
- (4) If you don't want extra spacing between sentences, say `\frenchspacing`. This will remain in effect until it is switched off by `\nonfrenchspacing`.

1. Fine-tuning text

1.1. Word and character spacing

- (1) T_EX inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.
- (2) T_EX interprets a period following a lowercase letter as the end of a sentence, **Prof. Svenonius** should be typeset as **Prof.**␣**Svenonius**.
- (3) An uppercase letter followed by a period is interpreted as not being at the end of a sentence, so \@ has to be inserted before the period to ensure extra spacing: **at CASTL.** is to be typeset as **at**␣**CASTL**␣␣.
- (4) If you don’t want extra spacing between sentences, say \frenchspacing. This will remain in effect until it is switched off by \nonfrenchspacing.
- (5) A small spacing is produced by the command \,. It might be useful when typesetting “‘left’ and ‘right’”: ‘\,’left’ and ‘right’,,’. If your source has ‘‘‘left’ and ‘right’’, it will come out as “‘left’ and ‘right’”.

1. Fine-tuning text

1.1. Word and character spacing

- (1) T_EX inserts more space between sentences than between words (according to traditional typesetting rules). So it inserts more space after ‘.’ and ‘!’ than after ‘,’. Since ‘.’ is used to end sentences as well as after abbreviations, this sometimes leads to confusion.
- (2) T_EX interprets a period following a lowercase letter as the end of a sentence, **Prof. Svenonius** should be typeset as **Prof._Svenonius**.
- (3) An uppercase letter followed by a period is interpreted as not being at the end of a sentence, so \@ has to be inserted before the period to ensure extra spacing: **at CASTL.** is to be typeset as **at_CASTL\@..**
- (4) If you don't want extra spacing between sentences, say \frenchspacing. This will remain in effect until it is switched off by \nonfrenchspacing.
- (5) A small spacing is produced by the command \,. It might be useful when typesetting “‘left’ and ‘right’”: ‘\,‘left’ and ‘right’\,’’. If your source has ‘‘‘left’ and ‘right’’, it will come out as “‘left’ and ‘right’”.

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

`\hspace*{length}` inserts the specified amount of space even when it's output at the beginning of a line.

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

`\hspace*{length}` inserts the specified amount of space even when it's output at the beginning of a line.

A blank before or after `\hspace` also appears in the output.

`Here_\hspace{1cm}1cm`

Here is 1cm

`Here_\is_\hspace{1cm}1cm`

Here is 1cm

`Here_\is_\hspace{1cm}_1cm`

Here is 1cm

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

`\hspace*{length}` inserts the specified amount of space even when it's output at the beginning of a line.

A blank before or after `\hspace` also appears in the output.

<code>Here_ is \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here_ is_ \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here_ is_ \hspace{1cm}_ 1cm</code>	Here is	1cm

(7) The length specification can also be negative, then it puts two items closer together.

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

`\hspace*{length}` inserts the specified amount of space even when it's output at the beginning of a line.

A blank before or after `\hspace` also appears in the output.

<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm

(7) The length specification can also be negative, then it puts two items closer together.

(8) `\hfill` is a command for inserting a 'rubber' length: it takes up all and only the amount of horizontal space available. (It's short for `\hspace{\fill}`, where `\fill` is a special 'rubber' length).

(6) Inserting horizontal space

`\hspace{length}` inserts the specified amount of blank space at the point where it appears in the text, but not at the beginning of an output line.

`\hspace*{length}` inserts the specified amount of space even when it's output at the beginning of a line.

A blank before or after `\hspace` also appears in the output.

<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm
<code>Here is \hspace{1cm} 1cm</code>	Here is	1cm

(7) The length specification can also be negative, then it puts two items closer together.

(8) `\hfill` is a command for inserting a 'rubber' length: it takes up all and only the amount of horizontal space available. (It's short for `\hspace{\fill}`, where `\fill` is a special 'rubber' length).

`1\hfill2` produces 1

2

`1\hfill2` produces 1

2

`1\hfill2\hfill3` produces 1

2

3

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

(9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

(9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

(9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hrulefill2\hrulefill3` gives 1_____2_____3

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

(9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hrulefill2\hrulefill3` gives 1_____2_____3

(10) `\quad` inserts the amount of horizontal space equal to the current typesize (e.g., 10 pt for a 10 pt font); `\qqquad` inserts the double.

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

- (9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hrulefill2\hrulefill3` gives 1_____2_____3

- (10) `\quad` inserts the amount of horizontal space equal to the current typesize (e.g., 10 pt for a 10 pt font); `\qqquad` inserts the double.

1.2. Vertical spacing

- (11) `\vspace{length}` inserts the specified amount of vertical space at the point where it appears in the text, but not at the beginning of a new page.

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

- (9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hrulefill2\hrulefill3` gives 1_____2_____3

- (10) `\quad` inserts the amount of horizontal space equal to the current typesize (e.g., 10 pt for a 10 pt font); `\qqquad` inserts the double.

1.2. Vertical spacing

- (11) `\vspace{length}` inserts the specified amount of vertical space at the point where it appears in the text, but not at the beginning of a new page.
`\vspace*{length}` inserts the specified amount of `vspace` even when it's output at the beginning of a page.

`1\hfill2` produces 1 2

`1\hfill2\hfill3` produces 1 2 3

- (9) `\dotfill` and `\hrulefill` work in the same way:

`1\dotfill2` gives 1.....2

`\dotfill\dotfill\dotfill text` gives text

`1\hrulefill2\hrulefill3` gives 1_____2_____3

- (10) `\quad` inserts the amount of horizontal space equal to the current typesize (e.g., 10 pt for a 10 pt font); `\qqquad` inserts the double.

1.2. Vertical spacing

- (11) `\vspace{length}` inserts the specified amount of vertical space at the point where it appears in the text, but not at the beginning of a new page.
`\vspace*{length}` inserts the specified amount of `vspace` even when it's output at the beginning of a page.

- (12) `\vfill` inserts a rubber length. `\vfill` `Some text` `\vfill` centers `Some text` vertically:

`Some text`

- (13) The commands `\bigskip`, `\medskip` and `\smallskip` insert vertical space depending on the fonts size.

- (12) `\vfill` inserts a rubber length. `\vfill` `Some text` `\vfill` centers `Some text` vertically:

`Some text`

- (13) The commands `\bigskip`, `\medskip` and `\smallskip` insert vertical space depending on the fonts size.

1.3. Line breaking

(14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.
- (16) A line break can be encouraged with `\linebreak[num]`, where *num* is a number between 0 and 4. *0* allows a line break where it normally wouldn't occur, such as in the middle of a word. `\linebreak[4]` is equivalent to `\linebreak`, compelling a line break. The line is justified.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.
- (16) A line break can be encouraged with `\linebreak[num]`, where *num* is a number between 0 and 4. *0* allows a line break where it normally wouldn't occur, such as in the middle of a word. `\linebreak[4]` is equivalent to `\linebreak`, compelling a line break. The line is justified.
- (17) `\nolinebreak` has the opposite effect; its syntax is identical to that of `\linebreak`.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.
- (16) A line break can be encouraged with `\linebreak[num]`, where *num* is a number between 0 and 4. *0* allows a line break where it normally wouldn't occur, such as in the middle of a word. `\linebreak[4]` is equivalent to `\linebreak`, compelling a line break. The line is justified.
- (17) `\nolinebreak` has the opposite effect; its syntax is identical to that of `\linebreak`.
- (18) `~` is a non-breakable space. `Section~1` makes sure *Section 1* appears in the same line.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.
- (16) A line break can be encouraged with `\linebreak[num]`, where *num* is a number between 0 and 4. *0* allows a line break where it normally wouldn't occur, such as in the middle of a word. `\linebreak[4]` is equivalent to `\linebreak`, compelling a line break. The line is justified.
- (17) `\nolinebreak` has the opposite effect; its syntax is identical to that of `\linebreak`.
- (18) `~` is a non-breakable space. `Section~1` makes sure *Section 1* appears in the same line.
- (19) Putting text in an `\mbox` has the same effect: `\mbox{$n$-tuple}` ensures *n-tuple* stays in the same line.

1.3. Line breaking

- (14) A new line can be forced with `\\` or `\newline`. The last line is left aligned.
- (15) With the optional argument `\\[length]`, you can specify space between the two lines. `\\*[length]` makes sure the two lines appear on the same page.
- (16) A line break can be encouraged with `\linebreak[num]`, where *num* is a number between 0 and 4. *0* allows a line break where it normally wouldn't occur, such as in the middle of a word. `\linebreak[4]` is equivalent to `\linebreak`, compelling a line break. The line is justified.
- (17) `\nolinebreak` has the opposite effect; its syntax is identical to that of `\linebreak`.
- (18) `~` is a non-breakable space. `Section~1` makes sure *Section 1* appears in the same line.
- (19) Putting text in an `\mbox` has the same effect: `\mbox{$n$-tuple}` ensures *n-tuple* stays in the same line.

1.4. Page breaking

(20) A page break is forced with `\newpage`.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.
- (22) Enclosing text in a `\vbox` makes sure all of it is typeset on the same page.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.
- (22) Enclosing text in a `\vbox` makes sure all of it is typeset on the same page.

1.5. Paragraph formatting

- (23) `\parindent` is the length by which the first lines of a paragraph are indented.
It can be modified by the length commands.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.
- (22) Enclosing text in a `\vbox` makes sure all of it is typeset on the same page.

1.5. Paragraph formatting

- (23) `\parindent` is the length by which the first lines of a paragraph are indented. It can be modified by the length commands.
- (24) `\parskip` is vertical spacing between paragraphs. Should be defined in `ex` or `em`. It's a rubber length: `\renewcommand{\parskip}{length_1pluslength_2minuslength_3}`.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.
- (22) Enclosing text in a `\vbox` makes sure all of it is typeset on the same page.

1.5. Paragraph formatting

- (23) `\parindent` is the length by which the first lines of a paragraph are indented. It can be modified by the length commands.
- (24) `\parskip` is vertical spacing between paragraphs. Should be defined in `ex` or `em`. It's a rubber length: `\renewcommand{\parskip}{length_1pluslength_2minuslength_3}`.
- (25) `\baselinestretch` is a number that magnifies the normal distance between baselines. `\renewcommand{\baselinestretch}{1.5}` enlarges line spacing by 50%. If given outside the preamble, it will only take effect when a new font size is declared.

1.4. Page breaking

- (20) A page break is forced with `\newpage`.
- (21) `\pagebreak` and `\nopagebreak` function just like `\linebreak` and `\nolinebreak`.
- (22) Enclosing text in a `\vbox` makes sure all of it is typeset on the same page.

1.5. Paragraph formatting

- (23) `\parindent` is the length by which the first lines of a paragraph are indented. It can be modified by the length commands.
- (24) `\parskip` is vertical spacing between paragraphs. Should be defined in `ex` or `em`. It's a rubber length: `\renewcommand{\parskip}{length_1pluslength_2minuslength_3}`.
- (25) `\baselinestretch` is a number that magnifies the normal distance between baselines. `\renewcommand{\baselinestretch}{1.5}` enlarges line spacing by 50%. If given outside the preamble, it will only take effect when a new font size is declared.

(26) `\noindent` suppresses indentation of the current paragraph; `\indent` ensures that it's indented.

- (26) `\noindent` suppresses indentation of the current paragraph; `\indent` ensures that it's indented.
- (27) Normally, the first paragraph of a section is not indented, not even with `\indent`. The package `indentfirst` changes this (just say `\usepackage{indentfirst}` in the preamble).

- (26) `\noindent` suppresses indentation of the current paragraph; `\indent` ensures that it's indented.
- (27) Normally, the first paragraph of a section is not indented, not even with `\indent`. The package `indentfirst` changes this (just say `\usepackage{indentfirst}` in the preamble).
- (28) The package `parskip` sets `\parindent` to zero and adds white space between paragraphs (and also modifies some of the parameters for lists).

- (26) `\noindent` suppresses indentation of the current paragraph; `\indent` ensures that it's indented.
- (27) Normally, the first paragraph of a section is not indented, not even with `\indent`. The package `indentfirst` changes this (just say `\usepackage{indentfirst}` in the preamble).
- (28) The package `parskip` sets `\parindent` to zero and adds white space between paragraphs (and also modifies some of the parameters for lists).

1.6. Page formatting

- (29) Pages in L^AT_EX are quite complex. You can get a glimpse with the package `layout`. The command `\layout` outputs a diagram of the current page.
- (30) The command `\enlargethispage{length}` adds `length` to the `\textheight` of the current page. `\enlargethispage*{length}` also shrinks interline spacing.

- (26) `\noindent` suppresses indentation of the current paragraph; `\indent` ensures that it's indented.
- (27) Normally, the first paragraph of a section is not indented, not even with `\indent`. The package `indentfirst` changes this (just say `\usepackage{indentfirst}` in the preamble).
- (28) The package `parskip` sets `\parindent` to zero and adds white space between paragraphs (and also modifies some of the parameters for lists).

1.6. Page formatting

- (29) Pages in L^AT_EX are quite complex. You can get a glimpse with the package `layout`. The command `\layout` outputs a diagram of the current page.
- (30) The command `\enlargethispage{length}` adds `length` to the `\textheight` of the current page. `\enlargethispage*{length}` also shrinks interline spacing.
- (31) If a document contains big figures or tables, T_EX will increase interline spacing to ensure every page is of the same height. To prevent this, say `\raggedbottom` in the preamble.

2. Tables

(32) Generalised table: `\begin{tabular}[pos]{col}`

$$\vdots$$

`\end{tabular}`

2. Tables

(32) Generalised table: `\begin{tabular}[pos]{col}`

⋮

`\end{tabular}`

(33) *pos* Vertical positioning of the table.

- t the top line of the table is aligned with the last line of text preceding the table.
 - b the bottom line of the table is aligned with the last line of text preceding the table.
- if no positioning is given, the table is centered on the last external line.

2. Tables

(32) Generalised table: `\begin{tabular}[pos]{col}`

`\end{tabular}`

(33) *pos* Vertical positioning of the table.

- t the top line of the table is aligned with the last line of text preceding the table.
 - b the bottom line of the table is aligned with the last line of text preceding the table.
- if no positioning is given, the table is centered on the last external line.

col The column-formatting argument. There must be an entry for each column of the table.

- l the text in the column is left-aligned.
- r the text in the column is right-aligned.
- c the text in the column is centered.

2. Tables

(32) Generalised table: `\begin{tabular}[pos]{col}`

`⋮`
`\end{tabular}`

(33) *pos* Vertical positioning of the table.

- t the top line of the table is aligned with the last line of text preceding the table.
 - b the bottom line of the table is aligned with the last line of text preceding the table.
- if no positioning is given, the table is centered on the last external line.

col The column-formatting argument. There must be an entry for each column of the table.

- l the text in the column is left-aligned.
- r the text in the column is right-aligned.
- c the text in the column is centered.

(34) Lines are terminated by `\\`, not `\cr`.

(35) Columns are separated by `&`.

2. Tables

(32) Generalised table: `\begin{tabular}[pos]{col}`

`⋮`
`\end{tabular}`

(33) *pos* Vertical positioning of the table.

- t the top line of the table is aligned with the last line of text preceding the table.
 - b the bottom line of the table is aligned with the last line of text preceding the table.
- if no positioning is given, the table is centered on the last external line.

col The column-formatting argument. There must be an entry for each column of the table.

- l the text in the column is left-aligned.
- r the text in the column is right-aligned.
- c the text in the column is centered.

(34) Lines are terminated by `\\`, not `\cr`.

(35) Columns are separated by `&`.

(36) The columns' width is unconstrained, so the table can extend beyond the margins.

(37) A very simple table:

	Name	Drink	Comments
1	Asbjørn	beer	might sing after 4 th pint!
2	Curt	gin & tonic	only Sapphire!
3	Patrik	Margarita	danger of karaoke
4	Sylvia	vodka & anything	preferably Stolichnaya

```
\begin{tabular}[t]{rlcl}  
  &Name      &Drink      &&Comments\\  
1  &Asbj\o{ }rn  &beer      &&&might sing...\\  
2  &Curt         &gin \& tonic  &&&only Sapphire!\\  
3  &Patrik       &Margarita    &&&danger of karaoke\\  
4  &Sylvia       &vodka \& anything  &&&preferably Stolichnaya\\  
\end{tabular}
```

(38) $\>\{decl\}$ inserts *decl* before a column, $\<\{decl\}$ inserts *decl* after a column (requires the package `array`).

- (38) `>\{decl\}` inserts *decl* before a column, `<\{decl\}` inserts *decl* after a column (requires the package `array`).

	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```
\begin{tabular}[t]{t}>\bfseries r>\itshape lcl}
  &Name          &Drink          &Comments\\
1  &Asbj\o{}rn    &beer          &might sing...\\
2  &Curt          &gin \& tonic  &only Sapphire!\\
3  &Patrik        &Margarita     &danger of karaoke\\
4  &Sylvia        &vodka \& anything &preferably Stolichnaya\\
\end{tabular}
```

- (39) `\hline` creates a horizontal line over the whole width of the table; `\cline{a-b}` creates a line over columns *a* to *b*. They can appear at the very beginning of a table or after `\\`.

- (39) `\hline` creates a horizontal line over the whole width of the table; `\cline{a-b}` creates a line over columns *a* to *b*. They can appear at the very beginning of a table or after `\\`.

	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```
\begin{tabular}[t]{t}{>{\bfseries}r>{\itshape}lcl}
\hline
  &Name          &Drink          &Comments\\ \hline
1  &Asbj\o{}rn    &&beer          &&might sing...\\ \cline{2-4}
2  &&Curt          &&gin \& tonic  &&only Sapphire!\\ \cline{2-4}
3  &&Patrik        &&Margarita     &&danger of karaoke\\ \cline{2-4}
4  &&Sylvia        &&vodka \& anything &&preferably Stolichnaya\\ \hline
\end{tabular}
```

(40) Vertical lines are specified in the table's preamble.

(40) Vertical lines are specified in the table's preamble.

	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Saphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```
\begin{tabular}[t]{t}{|>{\bfseries}r|>{\itshape}l|c|l|}  
\hline  
    &Name          &Drink          &Comments\\\hline  
1  &Asbj\o{}rn    &beer          &&might sing...\\\cline{2-4}  
2  &Curt          &&gin \& tonic  &&only Saphire!\\\cline{2-4}  
3  &Patrik        &&Margarita     &&danger of karaoke\\\cline{2-4}  
4  &Sylvia        &&vodka \& anything &&preferably Stolichnaya\\\hline  
\end{tabular}
```

(41) With the package `arydshln`, you can typeset tables with dashed and dotted lines.

(41) With the package `arydshln`, you can typeset tables with dashed and dotted lines.

	Vertical lines	Horizontal lines
predefined	:	<code>\hdashline</code> <code>\cdashline{a-b}</code>
user-defined	<code>:{dash/gap}</code>	<code>\hdashline[dash/gap]</code> <code>\cdashline{a-b}[dash/gap]</code>

(41) With the package `arydshln`, you can typeset tables with dashed and dotted lines.

	Vertical lines	Horizontal lines
predefined	:	<code>\hdashline</code> <code>\cdashline{a-b}</code>
user-defined	<code>:{dash/gap}</code>	<code>\hdashline[dash/gap]</code> <code>\cdashline{a-b}[dash/gap]</code>

The dashed lines can be redefined globally with `\dashlinedash` and `\dashlinegap` (default is 4 pt for both.)

(41) With the package `arydshln`, you can typeset tables with dashed and dotted lines.

	Vertical lines	Horizontal lines
predefined	:	<code>\hdashline</code> <code>\cdashline{a-b}</code>
user-defined	<code>;\dash/gap</code>	<code>\hdashline[dash/gap]</code> <code>\cdashline{a-b}[dash/gap]</code>

The dashed lines can be redefined globally with `\dashlinedash` and `\dashlinegap` (default is 4 pt for both.)

`\usepackage{arydshln}` must follow `\usepackage{array}`.

	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```

\begin{tabular}[t]{t}{|>\bfseries}r|>\itshape}l:c;{1pt/1pt}1||\hline
      &Name          &Drink          &Comments\\\hline\hline
1    &Asbj\o{}rn      &&beer          &&might sing...\\\cdashline{2-4}
2    &&Curt            &&gin \& tonic  &&only Sapphire!\\\cdashline{2-4}
3    &&Patrik          &&Margarita     &&danger of karaoke\\\cdashline{2-4}
4    &&Sylvia          &&vodka \& anything &&preferably...\\\cdashline{2-4}
\end{tabular}

```

(42) `\multicolumn{number}{col}{text}` combines the following *number* of columns into one column. It also resets all formatting for that column.

- (42) `\multicolumn{number}{col}{text}` combines the following *number* of columns into one column. It also resets all formatting for that column.

DRINKS			
	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```
\begin{tabular}[t]{|>{\bfseries}r|>{\itshape}l:c;{1pt/1pt}l|}\hline
\multicolumn{4}{c}{\textsc{Drinks}}\hline
&Name&&Drink&&Comments\\\hline
1&Asbjørn&&beer&&might sing...\\cdashline{2-4}
2&Curt&&gin \& tonic&&only Sapphire!\\cdashline{2-4}
3&Patrik&&Margarita&&danger of karaoke\\cdashline{2-4}
4&Sylvia&&vodka \& anything&&preferably Stolichnaya\\\hline
\end{tabular}
```

Vertical lines around `\multicolumns` have to be specified spearately.

Vertical lines around `\multicolumns` have to be specified separately.

DRINKS			
	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```

\begin{tabular}[t]{t}[>{\bfseries}r|>{\itshape}l:c;{1pt/1pt}l|]\hline
\multicolumn{4}{|c|}{\textsc{Drinks}}\\ \hline
&Name&&Drink&&Comments\\ \hline
1&Asbjørn&&beer&&might sing...\\ \cdashline{2-4}
2&Curt&&gin \& tonic&&only Sapphire!\\ \cdashline{2-4}
3&Patrik&&Margarita&&danger of karaoke\\ \cdashline{2-4}
4&Sylvia&&vodka \& anything&&preferably Stolichnaya\\ \hline
\end{tabular}

```

- (43) To have text in the table to be aligned with text outside the table, use `\firsthline` and `\lasthline` (requires the package `array`).

- (43) To have text in the table to be aligned with text outside the table, use `\firsthline` and `\lasthline` (requires the package `array`).

The table:

DRINKS			
	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolichnaya

```
The table: \begin{tabular}[t]{|>\bfseries r|>\itshape l:c;{1pt/1pt}l|}
\firsthline\multicolumn{4}{|c|}{\textsc{Drinks}}\\ \hline
      &Name      &&Drink      &&Comments\\ \hline
1  &Asbjørn    &&beer      &&might sing...\\ \cdashline{2-4}
2  &Curt       &&gin \& tonic  &&only Sapphire!\\ \cdashline{2-4}
3  &Patrik     &&Margarita  &&danger of karaoke\\ \cdashline{2-4}
4  &Sylvia     &&vodka \& anything  &&preferably Stolichnaya\\ \hline
\end{tabular}
```

- (44) The column-formatting argument `p{length}` produces columns of the specified width. Text in these columns is justified, and the first line of it is aligned with the first line of other columns in the row.
- `m{length}` vertically aligns text in the middle of the column, `b{length}` aligns text to the bottom of the column.

- (44) The column-formatting argument `p{length}` produces columns of the specified width. Text in these columns is justified, and the first line of it is aligned with the first line of other columns in the row. `m{length}` vertically aligns text in the middle of the column, `b{length}` aligns text to the bottom of the column.

DRINKS			
	<i>Name</i>	Drink	Comments
1	<i>Asbjørn</i>	beer	might sing after 4 th pint!
2	<i>Curt</i>	gin & tonic	only Sapphire!
3	<i>Patrik</i>	Margarita	danger of karaoke
4	<i>Sylvia</i>	vodka & anything	preferably Stolich- naya

```
\begin{tabular}[t]{|>{\bfseries}r|>{\itshape}l:c;{1pt/1pt}p{8em}|}\hline
\multicolumn{4}{|c|}{\textsc{Drinks}}\hline
      &Name      &&Drink      &&Comments\\\hline\hline
1  &Asbj\o{}rn  &&beer      &&\&might sing...\cdashline{2-4}
2  &Curt        &&gin \& tonic  &&\&only Sapphire!\cdashline{2-4}
3  &Patrik      &&Margarita   &&\&danger of karaoke\cdashline{2-4}
4  &Sylvia      &&vodka \& anything &&\&preferably Stolichnaya\\\hline
\end{tabular}
```

- (45) `@{text}` in the table preamble inserts *text* between two columns, and eliminates default intercolumn spacing. Declarations or `\hspace` can also be inserted.

- (45) `@{text}` in the table preamble inserts `text` between two columns, and eliminates default intercolumn spacing. Declarations or `\hspace` can also be inserted.

DRINKS			
	<i>Name</i> ⇒	Drink	Comments
1	<i>Asbjørn</i> ⇒	beer	might sing after 4 th pint!
2	<i>Curt</i> ⇒	gin & tonic	only Sapphire!
3	<i>Patrik</i> ⇒	Margarita	danger of karaoke
4	<i>Sylvia</i> ⇒	vodka & anything	preferably Stolich- naya

```

\begin{tabular}[t]{|>{\bfseries}r|>{\itshape}l{ $ \rightarrow$ }%
c;{1pt/1pt}{p{8em}}|\hline
\multicolumn{4}{|c|}{\textsc{Drinks}}\\ \hline
      &Name          &&Drink          &&Comments\\ \hline
1    &Asbjørn          &&beer          &&might sing...\\ \cdashline{2-4}
2    &Curt             &&gin \& tonic  &&only Sapphire!\\ \cdashline{2-4}
3    &Patrik           &&Margarita     &&danger of karaoke\\ \cdashline{2-4}
4    &Sylvia           &&vodka \& anything &&preferably Stolichnaya\\ \hline
\end{tabular}

```

- (46) Tables of a pre-specified width can be produced with the package `tabularx`. The width of columns having `X` as the formatting argument is adjusted to yield a table of the desired width. Extremely useful when typesetting tables with lots of text in some cells.

- (46) Tables of a pre-specified width can be produced with the package `tabularx`. The width of columns having `X` as the formatting argument is adjusted to yield a table of the desired width. Extremely useful when typesetting tables with lots of text in some cells.

Table preamble: `\begin{tabularx}{width}[pos]{col}`, where *width* is a length.

- (46) Tables of a pre-specified width can be produced with the package `tabularx`. The width of columns having `X` as the formatting argument is adjusted to yield a table of the desired width. Extremely useful when typesetting tables with lots of text in some cells.

Table preamble: `\begin{tabularx}{width}[pos]{col}`, where `width` is a length.

DRINKS			
	<i>Name</i> ⇒	Drink	Comments
1	<i>Asbjørn</i> ⇒	beer	might sing after 4 th pint!
2	<i>Curt</i> ⇒	gin & tonic	only Sapphire!
3	<i>Patrik</i> ⇒	Margarita	danger of karaoke
4	<i>Sylvia</i> ⇒	vodka & anything	preferably Stolichnaya

```
\begin{tabularx}{\textwidth-\labelwidth}%
{|>{\bfseries}r|>{\itshape}l@{ $\rightarrow$ }c;{1pt/1pt}X|}\hline
\multicolumn{4}{|c|}{\textsc{Drinks}}\\ \hline
&Name &&Drink &&Comments\\ \hline
1 &Asbj\o{}rn &&beer &&might sing...\cdashline{2-4}
2 &Curt &&gin \& tonic &&only Sapphire!\cdashline{2-4}
3 &Patrik &&Margarita &&danger of karaoke\cdashline{2-4}
4 &Sylvia &&vodka \& anything &&preferably Stolichnaya\\ \hline
\end{tabularx}
```

- (47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

- (47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

```
\newcolumntype{C}{>{$}c<{$}}
```

(47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

```
\newcolumntype{C}{>{$}c<{$}}
```

(48) Global modification of table parameters:

- `\tabcolsep` is half of the total spacing between columns – it is inserted before and after every column (the leftmost and rightmost one as well). It's a length, and can be manipulated accordingly.

(47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

```
\newcolumntype{C}{>{$}c<{$}}
```

(48) Global modification of table parameters:

- `\tabcolsep` is half of the total spacing between columns – it is inserted before and after every column (the leftmost and rightmost one as well). It's a length, and can be manipulated accordingly.
- `\arrayrulewidth` is the thickness of the horizontal and vertical lines in tables. Also a length.

(47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

```
\newcolumntype{C}{>{$}c<{$}}
```

(48) Global modification of table parameters:

- `\tabcolsep` is half of the total spacing between columns – it is inserted before and after every column (the leftmost and rightmost one as well). It's a length, and can be manipulated accordingly.
- `\arrayrulewidth` is the thickness of the horizontal and vertical lines in tables. Also a length.
- `\doublerulesep` is the distance between two `\hlines`. Also a length.

(47) With `\newcolumntype{type}{decl}`, one can define new column formatting arguments.

`\newcolumntype{C}{>{$}c<{$}}`

(48) Global modification of table parameters:

- `\tabcolsep` is half of the total spacing between columns – it is inserted before and after every column (the leftmost and rightmost one as well). It's a length, and can be manipulated accordingly.
- `\arrayrulewidth` is the thickness of the horizontal and vertical lines in tables. Also a length.
- `\doublerulesep` is the distance between two `\hlines`. Also a length.
- `\arraystretch` is the distance between table rows. This is a *factor*, it can be changed with `\renewcommand{\arraystretch}{num}` (*num* is a number, 1 by default).

Homework

Typeset the following table:

1.

/dɒgz/	SHARE[VOICE]	ID[VOICE]	*[VOICE]
a. [dɒgz]			**
b. [dɒks]		**	
c. [dɒgs]	*	*	*
d. [dɒkz]	*	*	*

- The table extends to the right margin if it is enclosed in an `\enumerate` environment.
- The last 3 columns are expandable.
- The lowercase letters are in a separate column, left aligned. They are produced automatically with a counter.
- There's no space between the first two columns of the table.