Markdown Syntax

This document describes the syntax of texts containing markdown elements. The markdown syntax is intended to simplify the writing of texts whose source is readable and can be easily formatted, e.g., as part of a web document. It is a subset of the original markdown syntax (e.g., internal links and pictures are missing) supported by the Curry package markdown.

Paragraphs and Basic Formatting

Paragraphs are separated by at least one line which is empty or does contain only blanks.

Inside a paragraph, one can *emphasize* text or also **strongly emphasize** text. This is done by wrapping it with one or two _ or * characters:

```
_emphasize_
*emphasize*
__strong__
**strong**
```

Furthermore, one can also mark `program code` text by backtick quotes (“`‘):

```
The function `fib` computes Fibonacci numbers.
```

Web links can be put in angle brackets, like in the link http://www.google.com:

```
<http://www.google.com>
```

If one wants to put a link under a text, one can put the text in square brackets directly followed by the link in round brackets, as in Google:

```
[Google](http://www.google.com)
```

If one wants to put a character that has a specific meaning in the syntax of Markdown, like * or _, in the output document, it should be escaped with a backslash, i.e., a backslash followed by a special character in the source text is translated into the given character (this also holds for program code, see below). For instance, the input text

```
\_word\_
```

produces the output “_word_”. The following backslash escapes are recognized:

```
\   backslash
`   backtick
*   asterisk
_   underscore
{}  curly braces
[]  square brackets
<>  angle brackets
()  parentheses
```

```
#   hash symbol
+   plus symbol
-   minus symbol (dash)
.   dot
    blank
!   exclamation mark
```

Lists and Block Formatting

An **unordered list** (i.e., without numbering) is introduced by putting a star in front of the list elements (where the star can be preceded by blanks). The individual list elements must contain the same indentation, as in

```
* First list element
  with two lines

* Next list element.

  It contains two paragraphs.

* Final list element.
```

This is formatted as follows:

- First list element with two lines
- Next list element.

  It contains two paragraphs.
- Final list element.

Instead of a star, one can also put dashes or plus to mark unordered list items. Furthermore, one could nest lists. Thus, the input text

```
 - Color:
     + Yellow
     + Read
     + Blue
 - BW:
     + Black
     + White
```

is formatted as

- Color:
  - Yellow
  - Read
  - Blue
- BW:
  - Black

– White

Similarly, **ordered lists** (i.e., with numbering each item) are introduced by a number followed by a dot and at least one blank. All following lines belonging to the same numbered item must have the same indent as the first line. The actual value of the number is not important. Thus, the input

```
1. First element


99. Second
    element
```

is formatted as

1. First element
2. Second element

A quotation block is marked by putting a right angle followed by a blank in front of each line:

```
> This is
> a quotation.
```

It will be formatted as a quote element:

> This is a quotation.

A block containing **program code** starts with a blank line and is marked by intending each input line by *at least four spaces* where all following lines must have at least the same indentation as the first non-blank character of the first line:

```
Here we see a program text:

    f x y = let z = (x,y)
            in (z,z)
```

The program code indentation is removed in the output so that this input is formatted as follows:

Here we see a program text:

```
f x y = let z = (x,y)
        in (z,z)
```

To visualize the structure of a document, one can also put a line containing only blanks and at least three dashes (stars would also work) in the source text:

```
-------------------------------------------------
```

This is formatted as a horizontal line:

Headers

The are two forms to mark headers. In the first form, one can "underline" the main header in the source text by equal signs and the second-level header by dashes:

```
First-level header
==================

Second-level header
-------------------
```

Alternatively (and for more levels), one can prefix the header line by up to six hash characters, where the number of characters corresponds to the header level (where level 1 is the main header):

```
# Main header

## Level 2 header

### Level 3

#### Level 4

##### Level 5

###### Level 6
```
