Functional Programming
Exercises Week 3
(for October 25, 2013)

Numbers in parentheses refer to the 6th edition of the course notes.
Exercises marked with ⋆ are optional and can be ignored.
Please have your `.ml` files accessible (e.g. on `zid-gpl.uibk.ac.at`).

1. Read Chapter 3 of the lecture notes.

2. (Exercise 3.1) Write the function `width : Picture.t -> int` as well as the function `height : Picture.t -> int` returning the width and height of a given picture, respectively.

3. (Exercise 3.2) Write functions for `Strng` to align l-strings within a box of given width. Three functions are needed:

```
left_justify : int -> t -> t
right_justify : int -> t -> t
center : int -> t -> t
```

In each case the integer argument denotes the width of the box. To illustrate the requirements consider the examples:

```
left_justify 5 ['H'; 'A'; 'L'] = ['H'; 'A'; 'L'; ' '; ' ']
right_justify 5 ['H'; 'A'; 'L'] = [' '; ' '; 'H'; 'A'; 'L']
center 5 ['H'; 'A'; 'L'] = [' '; 'H'; 'A'; 'L'; ' ']
```

Hint: You can ignore the case if a string is too long.

4. (Exercise 3.3) Write a function `paragraph : int -> string -> Picture.t` that—given the desired text width w and some text t—constructs a picture of width w such that the content of t is split into as many lines of text as needed to fit into a paragraph of w columns.

Hint: You may break lines at any character.

5. (Exercise 3.9) Caesar's cipher encodes a text by replacing each letter by another letter some fixed number (the *key*) of positions down the alphabet. E.g., encoding `HELLO` with a key of 2 yields `JGNNQ`.

a) Write a function `pos_of_char : char -> int` that returns the position in the alphabet associated with an (uppercase) letter, as well as the inverse of this function, i.e., `char_of_pos : int -> char`. E.g., `pos_of_char 'A'` yields `0` and the expression `char_of_pos 25` evaluates to `'Z'`.

Hint: The predefined functions `char_of_int` and `int_of_char` may be useful.

b) Implement the function `encode : int -> Strng.t -> Strng.t` and the function `decode : int -> Strng.t -> Strng.t` which encode and decode an l-string using a given key.

c) Write a function `crack : Strng.t -> Strng.t list` that returns the list of all possible decodings for a given ciphertext. Use this function to decipher the text `RHNVKTVDXWMAXVHWX`.

⋆. (Exercise 3.4) Modify `paragraph` from Exercise 4 in a way that it is possible to left-justify, right-justify, or center each line of text. Break lines at white space only (you may assume that no word is longer than the width of the paragraph. E.g., the function should output (approximately):

```
# Picture.paragraph2 12 Strng.right_justify "Functional programs \
  are readable like a book.";;
-␣:␣Picture.t␣=
␣␣Functional
programs␣are
␣␣␣␣readable
like␣a␣book.
```

Hint: Split the task into the following parts:

a) Separate words by white space

b) Combine words (such that they fit into a single line)

c) Align words (using Exercise 3)

d) Make the picture