Computer Architecture
Homework 3

We work with the following data type, which describes the type of the cells of a linked list of strings of characters:

```
struct cell{
  char *key;
  struct cell *next;
};
```

Exercise 1.

§1-1. Write a function

```
struct cell *cons(char *string, struct cell *list);
```

which appends a copy of the string `string` to the beginning of the list `list` and returns the result. Note that you should use the `malloc` function in order to copy the string.

§1-2. Write a function

```
void print_list(struct cell *list)
```

which prints the elements of a list, one per line.

§1-3. Write a function

```
int list_length(struct cell *list)
```

which returns the length of the list given as parameter.

As usual, write a function `main` and test the code which you have just written.

Exercise 2.

§2-1. Does the `main` program of the previous exercise free all the allocated memory?

§2-2. Write a function

```
free_list(struct cell *list)
```

which frees the memory allocated for a list, and modify your `main` so that it frees all the used memory.

§2-3. What happens if you call `free_list` twice on the same list?

Exercise 3.

§3-1. Write a function

```
struct cell *read_words(char *filename)
```

which opens the file `filename` in read mode, and stores every line which it contains in a linked list. Do not forget to close the file in the end.

§3-2. Write a function `main` which reads the file

```
/usr/share/dict/words
```

stores the content in a linked list, and then prints the length of it.

Exercise 4.

§4-1. Write a function

```
int stringcompare(char *stringone, char *stringtwo)
```

which returns true if the two strings `stringone` and `stringtwo` are equal, and false otherwise, without taking care of the difference between the letter case.

§4-2. Write a function

```
int list_member(char *string, struct cell *list)
```

which returns true if the string `string` is an element of the list `list`, using the function `stringcompare` implemented in §4.1.

Exercise 5.

§5. Write a spellchecker which reads the file `/usr/share/dict/words` and stores its content in a linked list. The spellchecker will then read words from the standard input and for each of them, indicate to the user whether the word is in the dictionary.

References and acknowledgments: among all the practice exercises in C which I found in the literature, I most enjoyed these exercises designed by my friend and colleague Juliusz Chroboczek, which I thus decided to translate and to adapt from French.