Computation 5EIA0/5AIA0
Interim Exam: String manipulation
15-10-2013, 13.45-15.30

In this interim exam you will develop a program that asks the user to input a sentence. The program will split this sentence into a sequence of words that are stored in a dictionary, it will sort these words on their length, and it will remove duplicate entries from the dictionary.

Task 1. Create a new project in Visual C++ with the name *IE2a-dictionary* and add a C++ file to the project. As always you should assign a meaningful name to the file (e.g., *dictionary.cpp*). Add a comment to the top of your file that includes your names and identity number. Your comment could look like the code shown below:

```
/**
 * Name: Sander Stuijk
 * Id: 667403
 */
```

Task 2. Create inside the `main` function an array `dict` with room for 100 character pointers and initialize all elements with the NULL value.

Task 3. Write a function "`int numberOfWords(char **dict)`" that counts the number of strings stored in the array. When you call this function with the empty array that you created in the previous task, the function should obviously return the value 0.

Task 4. Write a function "`void printWords(char **dict)`" that prints all strings stored in the array `dict`. When the dictionary contains no words, the function should print a message that the dictionary is empty.

When the dictionary contains the words "This", "is", and "story", you should see the following output:

```
Dictionary:
- This
- is
- story
```

When the dictionary is empty, you should see the following output:

```
Dictionary:
The dictionary is empty.
```

Task 5. Write a function "`void splitSentenceToWords(char **dict, char *sentence)`" that splits the string `sentence` into a sequence of words and that stores each word separately in the array `dict`.

You may assume that the sentence contains only letters and white space characters (i.e., it contains no digits or punctuations). Words in the sentence are separated with one (or more) white space characters.

When you enter the sentence "This is a story about a monkey and a banana" you should see the following output:

```
Enter a sentence: This is a story about a monkey and a banana
Dictionary:
- This
- is
- a
- story
- about
- a
- monkey
- and
- a
- banana
```

Note: When you do not succeed to write this function, then you should write a function that asks the user to input 10 words and that stores these 10 words as 10 separate elements in the array `dict`. You may then use this function and the content of the array `dict` in the remainder of this exam. Of course, you will receive less points for this task, but it will allow you to complete the remaining tasks.

Task 6. Write a function "`void swapWords(char **dict, int i, int j)`" that swaps the words at index `i` and `j` inside the dictionary contained in the array `dict`.

Task 7. Write a function "void `sortWordsOnSize(char **dict)`" that orders all words in the dictionary contained in the array `dict` based on their length.

When you enter the sentence "This is a story about a monkey and a banana", the output of your program should look as follows:

```
Enter a sentence: This is a story about a monkey and a banana
Dictionary:
- This
- is
- a
- story
- about
- a
- monkey
- and
- a
- banana
Sort dictionary on size.
Dictionary:
- a
- a
- a
- is
- and
- This
- story
- about
- monkey
- banana
```

Hint: Use the function `swapWords` that you wrote in the previous task.

Task 8. Write a function "`int countWord(char **dict, char *word)`" that returns the number of times the string `word` appears in the dictionary stored inside the array `dict`.

Task 9. Write a function "`void removeDuplicates(char **dict)`" that removes all duplicate entries from the dictionary (i.e., when a word appears multiple times in the dictionary all entries of this word except for one should be removed).

When you enter the sentence "This is a story about a monkey and a banana", the output of your program should look as follows:

```
Enter a sentence: This is a story about a monkey and a banana
Dictionary:
- This
- is
- a
- story
- about
- a
- monkey
- and
- a
- banana
Sort dictionary on size.
Dictionary:
- a
- a
- a
- is
- and
- This
- story
- about
- monkey
- banana
Remove duplicates.
Dictionary:
- a
- is
- and
- This
- story
- about
- monkey
- banana
```

Hint: Use the function `countWord` that you wrote in the previous task to check for duplicate entries of a word. Remember also to **free** the allocated memory that you no longer use.