

Computation I 5EIA0

Practice Exam: Dictionary (v1.2)

21 September 2020 18:30–20:30

In this exercise you will develop a dictionary program. The user will be able to add and remove words from the dictionary. In addition, the user can manually swap words in the dictionary or automatically sort all words in alphabetical order.

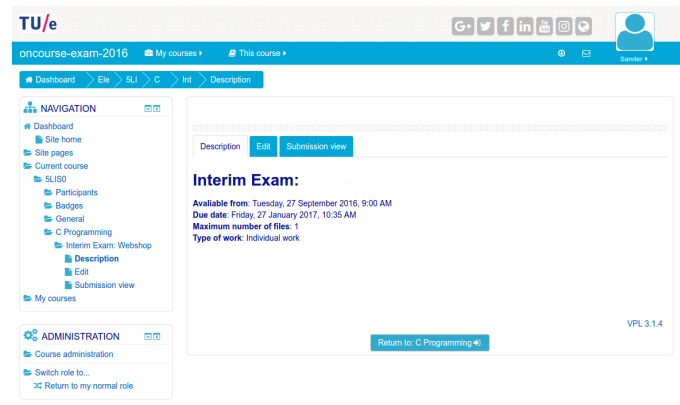
function	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	cases per fn	% per fn	cumulative %
quit	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	12%	12%
number of words			1		1	1											1	1	6%	18%
add word				1	1	1	1	1		1	1	1	1	1	1	1	1	3	18%	35%
print dictionary							1	1		1	1	1	1			1	1	1	6%	41%
remove word								1	1	1	1						1	4	24%	65%
swap words												1	1				1	2	12%	76%
largest word															1		1	1	6%	82%
sort dictionary															1	1	1	3	18%	100%

Figure 1: Test cases and points per task if correctly implemented. Note that you do not have to implement all functions for subsequent tasks. For example, if you do not implement e.g. number of words then you can still get a 7.5.

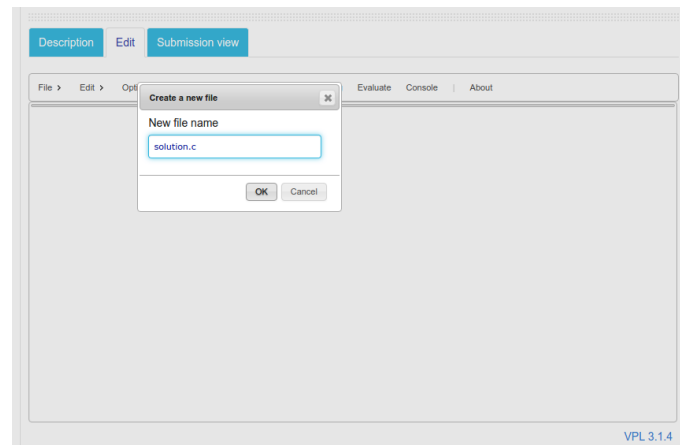
Important

- Your program is automatically submitted at the end of the exam.
- Your grade is based on the number of cases passed. So try to complete the cases one by one.
- You can press “evaluate” as often as you like during the exam to evaluate your solution. We advise you to do this regularly during the exam.
- You must use follow the instructions of the exam. For example, you may not use an array if a linked-list implementation is asked for.
- You are only allowed to use the Kernighan & Ritchie paper book in the Dutch or English language as a reference during the exam. No other electronic or printed material are allowed. An electronic version of the book is given out during the exam.
- The grade is based on your **last submission**. So make sure you submit a working version that completes as many cases as possible. It’s also wise not to make last-minute changes.
- Your program must work for all inputs, not just the test cases. For example, when a `#define MAX 10` must be defined as part of the exam, then your program should work for all values of MAX. We will change the test cases after the exam (but the number of test cases per task, i.e., the grade of correct programs will not change).

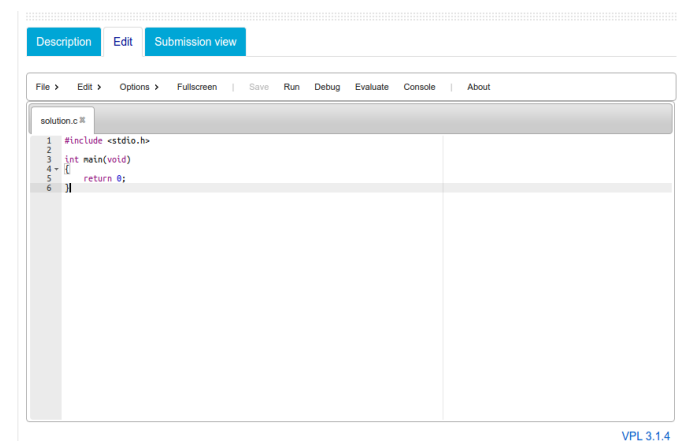
Task 1. Open the correct C programming exam on exam.oncourse.tue.nl. You will see a screen similar to:



Task 2.
Select the “Edit” tab and the following screen will appear:



Provide a useful name for your C file (e.g., solution.c) and press “Ok”.
You will then see the following screen:



You can write your own C program in the text editor that is now shown in your browser. Once you press “Save” you can “Run” and “Evaluate” your program. Using the “Run” command you will see a terminal where you can provide input to your program. You can use this to debug your program. Using the “Evaluate” command all test cases are evaluated and at the end the results are displayed (grade and errors if present).

Task 3. Write a C program that asks the user to select the command that needs to be performed. The commands that need to be supported are listed in the following table:

command	operation
q	quit program
n	number of words
a	add word
p	print dictionary
r	remove word
s	swap words
l	largest word
o	sort dictionary

In this task you only need to implement the quit command. In later tasks you will implement the remainder. Hint: make a loop in which you ask for a single character command, like this:

```
char cmd;
do {
    printf("Command [qnapsrlo]? ");
    scanf(" %c",&cmd); // notice the space before the %
    ...
} while (cmd != 'q');
```

Print the error message shown below if an invalid command is given:

```
Command [qnapsrlo]? A
Unknown command 'A'.
Command [qnapsrlo]? q
Bye!
```

Your program must produce the *exact* output, including all spaces, punctuation, capitalisation, quotes, etc.

Task 4. Inside the main function declare a two-dimensional array called dict with room for 10 strings of 30 characters each using the following #define:

```
#define MAXWORDS 10
#define MAXLENGTH 30
```

Initialize all elements with the empty string (hint: you can use = { { "" } }; to do this).

Write a function "int numberOfWordsInDict(char dict[MAXWORDS][MAXLENGTH])" that counts the number of non-empty strings stored in the array. When you call this function with the empty array that you created in the previous task, the function should return the value 0. In your main function you should call the function like this: numberOfWordsInDict(dict) where dict is the variable declared in your main program. Add the 'n' command to print the number of words in the dictionary. This is an example output:

```
Command [qnapsrlo]? n
The dictionary contains 0 words.
Command [qnapsrlo]? q
Bye!
```

Task 5. Write a function “void addWord(char dict[MAXWORDS][MAXLENGTH], char word[])” that adds the string word to the dictionary stored in the array dict. The function should add the word in the first empty position of the array. Hence, the first word that is added to the dictionary should be added at index 0 of the array dict. The next word needs to be added at index 1 of the array, etc. When the user tries to add more words to the dictionary than its maximal size, the function should not add the word to the dictionary. Instead it should print the *exact* error message stating “The dictionary is full.”.

Add the ‘a’ command to the main function such that the user can add a word to the dictionary. A word is any sequence of non-white-space characters and can be read using the “%s” format string for scanf. An example output is:

```
Command [qnprsl0]? a
Add a word: Apple
Command [qnprsl0]? a
Add a word: Pear
Command [qnprsl0]? q
Bye!
```

Hint: You can use the function numberOfWordsInDict to determine the index at which the word must be added in the array dict.

Hint: Use #include <string.h> to be able to use the string-manipulation functions. In particular, you can use the strcpy(dict[i],word) function to copy the string word into the array dict[i]. See Appendix B3 of the K&R book.

Task 6. Write a function “void printDict(char dict[MAXWORDS][MAXLENGTH])” that prints all strings stored in the array dict. Add the ‘p’ command in the main loop. When the dictionary contains no words, the function should print a message that the dictionary is empty.

Example output:

```
Command [qnprsl0]? p
Dictionary:
The dictionary is empty.
Command [qnprsl0]? a
Add a word: Apple
Command [qnprsl0]? p
Dictionary:
- Apple
Command [qnprsl0]? q
Bye!
```

Task 7. Write a function "void removeWord(char dict[MAXWORDS][MAXLENGTH], char word[])" that removes the first occurrence of the word word from the dictionary contained in the array dict. The function should ensure that words (i.e., strings) contained in the dictionary are placed at the first elements of the array. In other words, when an empty string is found at element i of the array, all following elements of the array should also be empty (i.e. contain the string ""). Ensuring that this property holds will make it easier to count the number of elements in the array and to sort these elements.

As the program output below illustrates, when you remove a word you should move the last word in the dictionary to the position that the removed word occupied. For example, see how when removing "Union" the last word "Orange" takes its place.

Use the command 'r' for the operation 'remove a word'. The output of your program should then look as follows, including the error message if the word is not in the dictionary:

```
Command [qnprsl0]? r
Remove a word: Apple
The word "Apple" is not in the dictionary.
Command [qnprsl0]? a
Add a word: Apple
Command [qnprsl0]? a
Add a word: Union
Command [qnprsl0]? a
Add a word: Apple
Command [qnprsl0]? a
Add a word: Orange
Command [qnprsl0]? p
Dictionary:
- Apple
- Union
- Apple
- Orange
Command [qnprsl0]? r
Remove a word: Union
Command [qnprsl0]? p
Dictionary:
- Apple
- Orange
- Apple
Command [qnprsl0]? a
Add a word: Banana
Command [qnprsl0]? p
Dictionary:
- Apple
- Orange
- Apple
- Banana
Command [qnprsl0]? r
Remove a word: Apple
Command [qnprsl0]? p
Dictionary:
- Banana
- Orange
- Apple
Command [qnprsl0]? q
Bye!
```

Task 8. Write a function “void swapWords(char dict[MAXWORDS][MAXLENGTH], char word1[], char word2[])” that swap the words word1 and word2 inside the dictionary contained in the array dict. When one or both words are missing in the dictionary, the function should print an error message.

Use the command ‘s’ for the operation ‘swap two words’. The output of your program should then look as follows:

```
Command [qnaprslo]? a
Add a word: Apple
Command [qnaprslo]? a
Add a word: Union
Command [qnaprslo]? a
Add a word: Apple
Command [qnaprslo]? a
Add a word: Orange
Command [qnaprslo]? p
Dictionary:
- Apple
- Union
- Apple
- Orange
Command [qnaprslo]? s
Swap two words:
Enter first word: Union
Enter second word: Orange
Command [qnaprslo]? p
Dictionary:
- Apple
- Orange
- Apple
- Union
Command [qnaprslo]? s
Enter first word: Union
Enter second word: Banana
Cannot swap words. At least one word is missing in the dictionary.
Command [qnaprslo]? q
Bye!
```

Task 9. Write a function “void largestWord(char dict[MAXWORDS] [MAXLENGTH])” that prints the largest word in the dictionary. If the dictionary is empty it prints nothing.

```
Command [qnprslol]? l
Command [qnprslol]? a
Add a word: Hello
Command [qnprslol]? l
The largest word in the dictionary is "Hello"
Command [qnprslol]? q
Bye!
```

Task 10. Write a function “void sortDict(char dict[MAXWORDS] [MAXLENGTH])” that orders all words in the dictionary contained in the array dict in alphabetical order.

Hint: The string compare function strcmp in the string.h library is very useful. See Appendix B3 of the K&R book.

Hint: The bubble sort algorithm that repeatedly compares two words and swaps them if necessary is probably the easiest way to sort the list, given that you already have the swapWords function. But you can also use the insertion sort, or even merge sort if you wish.

Of course you will have to extend the set of operations supported by your program. Use the command ‘o’ for the operation ‘sort dictionary’. The output of your program should then look as follows:

```
Command [qnprslol]? a
Add a word: Apple
Command [qnprslol]? a
Add a word: Union
Command [qnprslol]? a
Add a word: Apple
Command [qnprslol]? a
Add a word: Orange
Command [qnprslol]? p
Dictionary:
- Apple
- Union
- Apple
- Orange
Command [qnprslol]? o
Command [qnprslol]? p
Dictionary:
- Apple
- Apple
- Orange
- Union
Command [qnprslol]? q
Bye!
```

Submission: Your *last* submission will be automatically graded.