Programming in C II, Group A
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: JTSK-320112
February 22nd, 2018

Assignment 5 - Stacks and Queues

- The problems of this assignment must be solved in C.
- Your programs should have the input and output formatting according to the testcases listed after the problems.
- Your programs should consider the grading rules:
  https://grader.eecs.jacobs-university.de/courses/320112/2018_1gA/Grading-Criteria-C2.pdf

Problem 5.1 *A Stack of characters* (3 points)

Presence assignment, due by 18:30 h today

Implement a stack, which is able to hold maximum 10 characters using an array implementation. You need to implement the functions `... push(...), ... pop(...), ... empty(...)` for emptying the stack, and `... isEmpty(...)` for checking if the stack is empty.
Your program should consist of `stack.h` (`struct` definition and function declarations), `stack.c` (function definitions) and `teststack.c` (`main()` function) and should use the following `struct`:

```
struct stack {
   unsigned int count;
   char array[10];      // Container
};
```

Input structure

There are several commands that manipulate the stack.
These commands are:

- `1` followed by a character pushes the character into the stack,
- `2` pops the character on the top off the stack and prints it on the standard output,
- `3` empties the stack by popping one element after the other and printing them on the standard output,
- `4` quits the program.

Output structure

If an element is popped off the stack then the element is printed. Stack underflow and overflow should be detected and an informational message (either "`Stack Overflow`" or "`Stack Underflow`") should be printed on the screen. In these cases no stack operation takes place.
You can assume that the input will be syntactically correct.

Testcase 5.1: input

```
1
d
1
q
2
1
&
3
2
4
```

Testcase 5.1: output

```
Pushing d
Pushing q
Popping q
Pushing &
Emptying Stack & d
Popping Stack Underflow
Quit
```

Problem 5.2 *A Stack for Converting Numbers* (2 points)

Modify the stack implemented for **Problem 5.1** such that you can use it for converting a positive decimal number stored in a `long` into the hexadecimal representation of the number.
Upload again all files related to this problem (i.e., `stack.h`, `stack.c` and `convertingstack.c`).
You can assume that the input will be valid.

Testcase 5.2: input

```
843722134
```

Testcase 5.2: output

```
The hexadecimal representation is:
324A2D96
```

Problem 5.3 *Stack of strings* (3 points)

Modify the `struct` from **Problem 5.1** and write a program that works with a stack of strings. Keep in mind the functions `strcpy()`, `strcmp()` and `strcat()`.
Use the string stack to reverse a sentence word by word. When the word "`exit`" is entered then your program should end its execution.
You can assume that all letters are lowercase and no punctuation marks are contained. You can also assume that the words will not be longer than 15 *characters and the sentences will have a length such that you do not need more than* 10 *positions on the stack.*
Your program should consist of one header file and two .c files (i.e., `stack.h`, `stack.c` and `wordstack.c`).

Testcase 5.3: input

```
dogs like cats and cats like dogs
bob likes tomatos do not like bob
exit
```

Testcase 5.3: output

```
The reversed sentence is:
dogs like cats and cats like dogs
The reversed sentence is:
bob like not do tomatos likes bob
```

Problem 5.4 *Adding to the queue* (2 points)

Download the files:
`https://jgrader.de/courses/320112/c/queue.h`
`https://jgrader.de/courses/320112/c/queue.c`
`https://jgrader.de/courses/320112/c/testqueue.c`

Take a look at the three files and understand the source code. Modify the three files such that the queue will contain characters instead of integers and the commands for the queue are: `1` to add, `2` to delete, `3` to print and `4` to quit (the second and third one needed for a later problem). Extend the source code of `queue.c` by implementing the `enqueue()` function. Follow the hints given in the slides (see Lecture 5 & 6, page 16).
You can assume that the input will be valid.

Testcase 5.4: input

```
1
a
1
c
1
f
4
```

Testcase 5.4: output

```
add char: Putting a into queue
1 items in queue
Type 1 to add, 2 to delete, 4 to quit:
add char: Putting c into queue
2 items in queue
Type 1 to add, 2 to delete, 4 to quit:
add char: Putting f into queue
3 items in queue
Type 1 to add, 2 to delete, 4 to quit:
Bye.
```

Bonus Problem 5.5 *qsort with heterogeneous data* (2 points)

Write a program where you can enter an integer value `n` from the keyboard. Then create two dynamically allocated arrays with `n` elements. The first array should contain floats and the second one should contain structs with a name (will not be longer than 30 characters but the name may contain spaces) and an age for a person. The program should read the values for both arrays from the keyboard. Then your program should call `qsort` (from `stdlib.h`) to sort 1) the array of floats in decreasing order, 2) the array of structs in alphabetical order of the name, 3) the arrays of structs in increasing order of the age. The resulting arrays after the sorting should be printed on the screen as in the following testcase.
You can assume that the input will be valid.

Testcase 5.5: input

```
3
1.7
-8.78
0.67
John
18
Mary
17
Ken
21
```

Testcase 5.5: output

```
Decreasingly sorted floats:
1.700 0.670 -8.780
Alphabetically sorted structs (name):
John:18 Ken:21 Mary:17
Increasingly sorted structs (age):
Mary:17 John:18 Ken:21
```

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` without any warnings (You can use `gcc -Wall -o program program.c`). Insert suitable comments (not on every line ...) to explain what your program does.
- Please name the programs according to the suggested filenames (they should match the description of the problem) in Grader. Otherwise you might have problems with the inclusion of header files.
  Each program **must** include a comment on the top like the following:

```
/*
   JTSK-320112
   a5_p1.c
   Firstname Lastname
   myemail@jacobs-university.de
*/
```

- You have to submit your solutions via *Grader* at
  `https://cantaloupe.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to
  `k.lipskoch@jacobs-university.de` **with a subject line that begins with JTSK-320112**.
  It is important that you do begin your subject with the coursenumber, otherwise I might have problems to identify your submission.
- Please note, that after the deadline it will not be possible to submit any solutions. It is useless to send late solutions by mail, because they will not be accepted.

This assignment is due by Tuesday, February 27th, 10:00 h.