Programming in C II, Group B
Jacobs University Bremen
Dr. Kinga Lipskoch
Xu He

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: http://minds.jacobs-university.de/sites/default/files/uploads/teaching/CProgrammingSpring18/Grading-Criteria-C2.pdf

Problem 5.1 *A stack of integers* (3 points)

Presence assignment, due by 18:30 h today
Implement a stack, which is able to hold maximum 12 integers 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;
  int array[12];        // Container
};
```

Input structure

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

- `s` followed by a number pushes the number into the stack,
- `p` pops a number on the top off the stack and prints it on the standard output,
- `e` empties the stack by popping one element after the other and printing them on the standard output,
- `q` quits the execution of 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 operation takes place.

You can assume that the input will be correct. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.1: input

```
s
5
s
7
p
s
3
e
p
q
```

Testcase 5.1: output

```
Pushing 5
Pushing 7
Popping 7
Pushing 3
Emptying Stack 3 5
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 an `unsigned int` into the binary representation of the number using division by 2 and storing the remainder of the division by 2 in the stack.
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. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.2: input

```
75
```

Testcase 5.2: output

```
The binary representation of 75 is 1001011.
```

Problem 5.3 *A stack of words* (3 points)

Modify the `struct` from **Problem 5.1** and write a program that tests a stack of words (the words will not be longer than 30 characters). Keep in mind the functions `strcpy()`, `strcmp()` and `strcat()`.
Use the word stack to check if a sentence (assume that the words are separated by spaces, all letters are lowercase and no punctuation marks are contained) is palindromic by words. For example, the sentence "`dogs like cats and cats like dogs`" is palindromic by words, because it reads the same from backwards (word by word). The program should terminate its execution if "`exit`" is entered as a sentence.
Your program should consist of one header file and two `.c` files (i.e., `stack.h`, `stack.c` and `wordstack.c`).
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.3: input

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

Testcase 5.3: output

```
The sentence is palindromic by words!
The sentence is not palindromic by words!
```

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

Download the files:
`https://grader.eecs.jacobs-university.de/courses/320112/c/queue.h`
`https://grader.eecs.jacobs-university.de/courses/320112/c/queue.c`
`https://grader.eecs.jacobs-university.de/courses/320112/c/testqueue.c`

Take a look at the three files and understand the source code. Extend the code of `queue.c` by implementing the `enqueue()` function. Follow the hints given in the slides (see Lecture 5 & 6, slide 16).
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 5.4: input

```
a
3
a
5
a
7
q
```

Testcase 5.4: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, q 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://grader.eecs.jacobs-university.de`.
  If there are problems (but **only** then) you can submit the programs by sending mail to `x.he@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.