Programming in C and C++
Jacobs University Bremen
Dr. Kinga Lipskoch

Course: CH-230-A
October 15th, 2019

Assignment 7 - Multiple Sources, Linked Lists, Function Pointers, Stacks

- The problems of this assignment must be solved in C or C++ (instruction in each problem).
- 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/ch_230_a/2019_2/Grading-Criteria-C-C++.pdf

Problem 7.1 *Multiple sources* (1 point)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C

Modify your solution for **Problem 6.8** such that you separate your source code into three files: struct declaration and function declarations in `linked_list.h`, function definitions in `linked_list.c`, and your `main` function in `use_linked_list.c`. Use exactly the provided names for your files.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.1: input

```
b
2
b
3
a
4
p
r
p
q
```

Testcase 7.1: output

```
3 2 4
2 4
```

Problem 7.2 *A doubly linked list of characters* (4 points)

Due by Monday, October 21st, 23:00 **Graded manually**
Language: C

Create a data type that implements a doubly linked list of characters. Write a program that tests your double linked list with the testcase given below. An integer value `1` entered from the keyboard will add the following character to the list to the beginning of the list, while a `2` will remove all elements with the given character from the list, a `3` will print the current list, while a `4` will print the elements of the list backwards. A `5` will empty the list, free the memory used by the doubly linked list and quit the execution of the program.
You can assume that the input does not contain logical errors (i.e., if the command is `2`, you can assume that a character will follow). However, a character to be deleted may not be currently in the list. In this case, the message "`The element is not in the list!`" should be printed on the standard output.
Use a switch-case statement to decide which action to take.
You can assume that the input will be valid regarding the structure. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.2: input

```
1
r
1
a
1
d
3
1
a
1
x
1
r
1
x
3
2
x
3
4
2
b
5
```

Testcase 7.2: output

```
d a r
x r x a d a r
r a d a r
r a d a r
The element is not in the list!
```

Problem 7.3 *Makefile* (2 points)

Due by Monday, October 21st, 23:00 **Graded manually**
Language: C

Continue with your solution for **Problem 7.1** in the following manner: write and upload a makefile called "`Makefile.txt`" which has multiple targets and can be used to: 1) generate all object files corresponding to the previous source files, 2) generate executable code from the object files and 3) delete all generated object files and the executable.
Submit the three source files and the makefile called "`Makefile.txt`".
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Problem 7.4 *Simple function pointers* (2 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**
Language: C

Write a program that reads a string and then repeatedly reads a command (one character) from the standard input.
If you press '`1`', then the string is printed uppercase on the standard output.
If you press '`2`', then the string is printed lowercase on the standard output.
If you press '`3`', then lowercase characters are printed uppercase and uppercase characters are printed lowercase on the standard output.
If you press '`4`', then the program should quit the execution.
Your `main` function (where you read the commands) or your other functions may not contain any `if` or `switch` statements for mapping the command to which function should be called. Your `main` function should contain an endless `while` loop.
Implement the solution using a function pointer array. The original string should not be changed.
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 7.4: input

```
This is a String
1
2
3
2
1
4
```

Testcase 7.4: output

```
THIS IS A STRING
this is a string
tHIS IS A sTRING
this is a string
THIS IS A STRING
```

Problem 7.5 *Quicksort with function pointers* (2 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**

Language: C

Write a program that sorts an array of `n` integers. After reading `n` and the values of the array from the standard input, the program reads a character and if this character is `'a'` then the sorting should be ascending, if the character is `'d'` then the sorting should be descending and if the character is `'e'` then the program should quit execution.

Your `main` function should contain an endless `while` loop for getting repeated input.

Your program should use function pointers and for sorting you should use the function `qsort` from `stdlib.h`.

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

Testcase 7.5: input

```
5
2
4
1
5
3
d
a
e
```

Testcase 7.5: output

```
5 4 3 2 1
1 2 3 4 5
```

Problem 7.6 *Bubblesort with function pointers* (4 points)

Due by Monday, October 21st, 23:00 **Graded manually**

Language: C

Write a program that reads an array of the following structure and sorts the data in ascending order by name or age using the *bubblesort algorithm*.

```
struct person {
        char name[30];
        int age;
};
```

Your program should read the number of persons from the standard input followed by the array of data corresponding to the persons. You should print the lists of sorted persons in ascending order with respect to their name (alphabetical order) and with respect to their age. Within the sorting according to age, note that if multiple persons have the same age, then they should be sorted alphabetically with respect to their name. Within the sorting according to name, note that if multiple persons have the same name, then they should be sorted with respect to their age.

Instead of writing two sorting functions use function pointers such that you can implement one `bubblesort` function able to sort according to different criteria.

You can assume that the input will be valid and that the names will not contain spaces. To pass the testcases your output has to be identical with the provided ones.

The pseudocode of the bubblesort algorithm is the following:

```
repeat
  swapped = false
  for i = 1 to length(A) - 1 inclusive do:
  /* if this pair is out of order */
    if A[i-1] > A[i] then
      /* swap them and remember something changed */
      swap( A[i-1], A[i] )
      swapped = true
    end if
  end for
until not swapped
```

Testcase 7.6: input

```
3
anne
23
mary
18
bob
20
```

Testcase 7.6: output

```
{anne, 23}; {bob, 20}; {mary, 18};
{mary, 18}; {bob, 20}; {anne, 23};
```

Problem 7.7 *A stack of integers* (3 points)

Due by Monday, October 21st, 23:00 **Graded automatically with testcases only**

Language: C

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 7.7: input

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

Testcase 7.7: output

```
Pushing 5
Pushing 7
Popping 7
Pushing 3
Emptying Stack 3 5
Popping Stack Underflow
Quit
```

How to submit your solutions

- Your source code should be properly indented and compile with `gcc` or `g++` depending on the problem without any errors or warnings (You can use `gcc -Wall -o program program.c` or `g++ -Wall -o program program.cpp`). 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:

```
/*
   CH-230-A
   a7_p1.[c or cpp or h]
   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 CH-230-A**.
  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 Monday, October 21st, 23:00.