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

Course: CH-230-A
October 22nd, 2019

Assignment 8 - Queues, Stacks, Files

- 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 8.1 *Adding to the queue* (2 points)

Presence assignment, due by 11:00 AM today **Graded automatically with testcases only**
Language: C
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 Tutorial 8, slide 12).
You can assume that the input will be valid. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.1: input

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

Testcase 8.1: 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.
```

Problem 8.2 *Removing from the queue* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**
Language: C
Extend the source code of `queue.c` from **Problem 8.1** by implementing the `dequeue()` function. Follow the hints given in the slides (see Tutorial 8, slide 13) and consider the case of a queue underflow.
You can assume that the input will be valid except the semantical possibility of reaching queue underflow. To pass the testcases your output has to be identical with the provided ones.

Testcase 8.2: input

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

Testcase 8.2: 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:
Removing 3 from queue
2 items in queue
Type a to add, d to delete, q to quit:
Removing 5 from queue
1 items in queue
Type a to add, d to delete, q to quit:
Bye.
```

Problem 8.3 *Printing the queue* (2 points)

Due by Monday, October 28th, 23:00 **Graded automatically with testcases only**

Language: C

Extend the source code of `queue.h`, `queue.c` and `testqueue.c` from **Problem 8.2** by adding and implementing the additional function `printq()` for printing the elements of the queue separated by spaces. If you enter '`p`', then the program should print the elements of the queue. Make sure that you can print more than once.

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

Testcase 8.3: input

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

Testcase 8.3: output

```
add int: Putting 3 into queue
1 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 5 into queue
2 items in queue
Type a to add, d to delete, p to print, q to quit:
add int: Putting 7 into queue
3 items in queue
Type a to add, d to delete, p to print, q to quit:
content of the queue: 3 5 7
3 items in queue
Type a to add, d to delete, p to print, q to quit:
Bye.
```

Problem 8.4 *A stack for converting numbers* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Modify the stack implemented for **Problem 7.7** 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 8.4: input

```
75
```

Testcase 8.4: output

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

Bonus Problem 8.5 *A stack of words* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Modify the `struct` from **Problem 8.4** 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 8.5: input

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

Testcase 8.5: output

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

Problem 8.6 *Read chars and write an int* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the first two characters from the file "`chars.txt`" and writes the sum of their ASCII code values as a number into "`codesum.txt`". Use an editor to create the input file "`chars.txt`". Your program is responsible to create the output file "`codesum.txt`".

You can safely assume that the content of the input file will be valid.

Problem 8.7 *Read and write doubles* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the keyboard the names of two files containing two double numbers. Your program should read these two values from the two files, compute their sum, difference, product and division, and write the results on separate lines into the file "`results.txt`".

You can safely assume that the input is valid, the two input files exist and each contains one valid double value.

Bonus Problem 8.8 *Merge two files* (1 point)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the content of two files "`text1.txt`" and "`text2.txt`" line by line and merges them into another file called "`merge12.txt`".

You can safely assume that the input is valid.

Problem 8.9 *Counting words in a file* (2 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads the content of a file given as input and counts the number of the words in the file. It is assumed the words are separated by one or multiple of the following characters: `' ' ',' '?' '!' '.' '\t' '\r' '\n'`.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/words.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/words2.txt`

You can assume that the content of the input file will be valid if existing.

Testcase 8.9: input

```
words.txt
```

Testcase 8.9: output

```
The file contains 17 words.
```

Bonus Problem 8.10 *"Authentification" with files* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the standard input a username and a password, and prints a message on the screen for granting or denying access depending on the fact if the user and the corresponding password are listed in a file which is given as input to the program.

The program should repeatedly check the username and its password introduced from the standard input until the word "`exit`" is introduced for the username. It is assumed that the word "`exit`" is not contained in the input file.

Do not store the whole information (list of usernames and passwords) in the main memory or do not read the whole information from the file for every request, but store only partial information (e.g., the usernames) and use the functions `ftell()` and `fseek()` to read the rest of the needed information.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/users.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/users2.txt`

You can assume that the content of the input file will be valid if existing.

Testcase 8.10: input

```
users.txt
ben
12b43
kate
jfd45
bill
iu3556
exit
```

Testcase 8.10: output

```
Access to user ben is granted.
Access to user kate is denied.
Access to user bill is granted.
Exiting ...
```

Problem 8.11 *Concat `n` files* (3 points)

Due by Monday, October 28th, 23:00 **Graded manually**

Language: C

Write a program which reads from the standard input the value of an integer `n` and then the names of `n` files. The program should concatenate the content of the `n` files separated by `'\n'` and write the result on the standard output and also into `output.txt`.

Read the input files and write the output file using the binary mode. Use a char buffer of size 64 bytes and chunks of size 1 byte when reading and the same buffer with chunks of size 64 bytes (or less if the last write and file size is not a multiply of 64) when writing.

For testing your solution to this problem, please use:

`https://grader.eecs.jacobs-university.de/courses/320112/c/file1.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/file2.txt`

`https://grader.eecs.jacobs-university.de/courses/320112/c/file3.txt`

You can assume that the content of the input files will be valid if existing.

Testcase 8.11: input

```
3
file1.txt
file2.txt
file3.txt
```

Testcase 8.11: output

```
Concating the content of 3 files ...
The result is:
The first file's
content.
The second file's content.
The
third files's
content.
The result was written into output.txt
```

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
   a8_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 28th, 23:00.