CS 211 Data Structures and Algorithms Lab
Aug -- November, 2019
Assignment 1
Total Marks: 10
Due on 24th August (Saturday)

The objective of this assignment is to implement Stack (Task 1), Queue (Task 2), and Doubly linked list (Task 3). Please read this document carefully before start coding.

Command-line arguments
Your program should receive two command line arguments: a file name followed by a number. For example, ‘./a.out input.txt 1000’ will be a typical execution of your program.

Input file
The input file will be a text file where each line will start with either of the four symbols: ‘+’, ‘-’, ‘?’, ‘!’. Here, ‘+’ stands for PUSH/ENQUEUE/INSERT, ‘-’ stands for POP/DEQUEUE/DELETE, ‘?’ stands for SEARCH, and ‘!’ stands for display. Additionally, ‘+’, ‘?’, and ‘-’ are followed by a positive integer, which is the argument for that particular operation. Note that the number followed by ‘-’ should be ignored for Task 1 (Stack) and Task 2 (Queue), as it is not relevant.

The content of an example input file is given below.

```
+ 10
+ 24
+ 1
!
- 24
- 1
? 10
? 9
!
```

Task 1 (Stack; 3 marks)
Implement a Stack using arrays. The size of the Stack (the number of elements that can be stored in the Stack) is given by the second command-line argument.

Output: The output of this task should be in a file named ‘stack.txt’. For every line of input file there should be a corresponding line in the output file.

- If there is a line with ‘+ y’ then your program should PUSH (if possible) the number ‘y’ into the Stack and then print ‘pushed y’ or ‘overflow’ (whichever is applicable) to the output file.

- If there is a line with '-' in the input file then your program should POP (if possible) an element from the Stack and print 'popped y' or 'underflow' (whichever is applicable) into the output file.
- A line with '? y' should cause your program to check if y is there in the Stack and print "found y" or "not found y" accordingly into the output file.
- A line with '!' should cause your program to print (last-in first) the content of the Stack into the output file.

For example, the stack.txt should contain the following lines when your program is invoked with the given input and stack-size 5:

pushed 10
pushed 24
pushed 1
1 24 10
popped 1
popped 24
found 10
not found 9
10

Task 2 (Queue; 3 marks)
Implement a Queue using arrays. The size of the Queue (the number of elements that can be stored in the Queue) is given by the second command-line argument.

Output: The output of this task should be in a file named 'queue.txt'. For every line of input file there should be a corresponding line in the output file.

- If there is a line with '+ y' then your program should ENQUEUE (if possible) the number 'y' into the Queue and then print 'enqueued y' or 'overflow' (whichever is applicable) to the output file.
- If there is a line with '-' in the input file then your program should DEQUEUE (if possible) an element from the Queue and print 'dequeued y' or 'underflow' (whichever is applicable) into the output file.
- A line with '? y' should cause your program to check if y is there in the Queue and print "found y" or "not found y" accordingly into the output file.
- A line with '!' should cause your program to display (first-in first) the content of the Queue.

For example, the queue.txt should contain the following lines when invoked with the given input and second command-line argument 5:

enqueued 10
enqueued 24

enqueued 1
10 24 1
dequeued 10
dequeued 24
not found 10
not found 9
1

Task 3 (Doubly Linked List, 4 marks):
Implement a Doubly Linked List (DLL). There is no size limit for the DLL. So, the second command-line argument is ignored for this task.

Output: The output of this task should be in a file named 'dll.txt'. For every line of input file there should be a corresponding line in the output file.

- If there is a line with '+ y' then your program should INSERT the number 'y' into the DLL and then print 'inserted y' to the output file.
- If there is a line with '- y' in the input file then your program should DELETE (if possible) the first occurrence (while searching from list head) of y from the DLL and print 'deleted y' or 'cannot delete y' (whichever is applicable) into the output file.
- A line with '? y' should cause your program to check if y is there in the DLL and print "found y" or "not found y" accordingly.
- A line with '!' should cause your program to display (last-in first) the content of the DLL.

For example, the dll.txt should contain the following lines when invoked with the given input:

inserted 10
inserted 24
inserted 1
1 24 10
deleted 24
deleted 1
found 10
not found 9
10

Submission

- The program you submit should output statck.txt, queue.txt and dll.txt when run (There should be only one main program to handle all the three tasks - of course, the source code can contain multiple files).
- The main file of your program should be named as main.<extension>, where the extension depends on the language you choose (Usage of C/C++ is mandatory for this assignment).

- Test well before submission. We have some hidden inputs with us to test your program. The mark you obtain is purely based on whether your program correctly gives outputs for the hidden inputs.
- If your program has only a single source file, please submit the file as it is. If your program has multiple source files, please submit your code as a zip file where the name of the zip file should be your roll number. It is important that you follow the input/output conventions exactly (including the naming scheme) as we may be doing an automated evaluation. There will be a penalty of 10% (on the mark you deserve otherwise) if you do not follow the naming conventions exactly.
- Follow some coding style uniformly. Provide proper comments in your code.
- Submit only through moodle. Submit well in advance. Any hiccups in the moodle/internet at the last minute is never acceptable as an excuse for late submission. Submissions through email will be ignored.
- Acknowledge the people (other than the instructor and TA) who helped you to solve this assignment. The details of the help you received and the names of the people who helped you (including internet sources, if applicable) should come in the beginning of the main file as a comment. Copying others' programs is a serious offence and deserving penalty will be imposed if found.

Evaluation

- To consider for first evaluation without penalty, you have to submit your program by 24th August (11:59 pm). If you submit after 24th August but on or before 31st August (11:59 pm), there will be a penalty of 10% on the marks you deserve otherwise.
- If you do not submit by 31st August, your program will not be considered for the first evaluation.
- We will do the first evaluation after 31st August. The marks you obtain will be proportional to the number of correct lines in the output files. We will use the 'diff' program to check the differences between the correct output file and the output file generated by your program. So, you may verify the correctness of output file by using diff program with sample output file before submission. (See the man page of diff for more info).
- After the first evaluation, you will get a chance to improve your program. For this, after modification, you can submit your code for second evaluation. It comes with a 20% penalty. The due date for the second evaluation will be announced after the first evaluation. Those who submit their code after 31st August and before the due date for second evaluation will also be considered for the second evaluation. Submission done after the due date of the second evaluation will be ignored.