CS 211 Data Structures and Algorithms Lab
Spring, 2022-23

Assignment	Mid semester
Objective	To implement Stack, Queue and Doubly Linked List
Total marks	15
Start date/time	9th June (Friday) 5:00 PM
Dead Line (Submission after deadline will not be evaluated)	10th June (Saturday) 5:00 PM
Penalty for violating naming convention(s)	10%

The objective of this assignment is to implement :
(i) ***Task 1:*** A Stack using Single Linked List;
(ii) ***Task 2:*** A Queue using Single Linked List;
(ii) ***Task 3***: A Doubly Linked List

Please read this document carefully before starting coding.

Command-line argument:
Your program should receive two command line arguments: a file name followed by a number.

For example, ***'./a.out input.txt 25'*** 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: '+', '-', '?', '!', where:

- '+' stands for PUSH/ENQUEUE/INSERT
- '-' stands for POP/DEQUEUE/DELETE
- '?' stands for SEARCH
- '!' stands for DISPLAY

Notes:
(i) '+', '?', and '-' are followed by a positive integer, which is the argument for that particular operation

(ii) the number followed by '-' should be ignored for Task 1 (Stack) and Task 2 (Queue), as it is not relevant

Task 1: Implement a Stack using Linked List

- *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 the 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 following outline shows the output file *stack.txt* corresponding to the input.txt file provided through CLI. The stack-size is set 5.

Input.txt	***stack.txt***
+ 10	pushed 10
+ 24	pushed 24
+ 1	pushed 1
!	1 24 10
- 24	popped 1
- 1	popped 24
? 10	found 10
? 9	not found 9
!	10

Task 2 : Implement a Circular Queue using singly Linked List

- *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 the 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 i.e 5:

Input.txt	*queue.txt*
+ 10	enqueued 10
+ 24	enqueued 24
+ 1	enqueued 1
!	10 24 1
- 10	dequeued 10
- 24	dequeued 24
? 10	not found 10
? 9	not found 9
!	1

Task 3 : 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 the 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:

Input.txt	*dll.txt*
+ 10	inserted 10
+ 24	inserted 24
+ 1	inserted 1
!	1 24 10
- 24	deleted 24
- 1	deleted 1
? 10	found 10
? 9	not found 9
!	10

Submission:

- Submit a valid **.c** file. [**.cpp** and **.txt** files as source code are not accepted.]
- Test with **gcc** compiler before submitting your code.
- The program you submit should output ***statck.txt, queue.txt*** *&* ***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 after your IIT Dharwad roll number. For example, ***<roll_no>.c***, where ***roll_no*** specifies your IIT Dharwad roll number (220010001.c)
    - Do the stress test of your program well before submission.
    - You may use the attached sample input files for testing, the corresponding output files are also attached.
    - We have some hidden inputs with us to test your program. *The marks you obtain are purely based on whether your program correctly gives outputs for the hidden inputs.*
- Your program should be a single source file, please submit the file as it is. As we are running an automated evaluation, it **is important that you must follow the input/output conventions exactly** (including the naming scheme). If you do not follow, ***there will be a penalty of 10% (on the mark you deserve).***
- Follow some coding style uniformly. Provide proper comments in your code.
- *Submit only through moodle*. ***Submit well in advance***. Any hiccups in the moodle at the last minute is never acceptable as an excuse for late submission. Submissions through email or any other means 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 and allowing others to copy your program are serious offenses and a deserving penalty(100%) will be imposed if found.***
- To be considered for the evaluation, you have to submit your program by the above mentioned deadline. ***No single minute relaxation on submission.***