Assignment 2 (Given: Nov 8, Due: Nov 17)

Write programs to solve the following problems using **stacks** and **queues**:

1. Write a program (utility) which tells whether the brackets in a file (given as input to the program) are balanced or not. In case the brackets in the file are balanced, the program should say that the brackets are balanced, otherwise, it should tell the line number on which the brackets are not balanced and the reason that why the brackets in the input file are not balanced? The program should take a file name as command line argument, e.g,. if the your program is `brackets.exe`, it should be able to check the balance of brackets in an file named `input.txt` as follows: `brackets.exe input.txt`. We should be able to give any file name as a command line argument. Use STL based stack.
2. Write a program which takes an expression in postfix notation as command line argument and evaluate it. For example, if the program is `postfix-eval.exe`, running `postfix-eval.exe "1 2 3 + -"` should display `-4`. The operands and operators should be separated by spaces. Operands may consist of 1 or more digits. The input should be treated as a single command line argument (use quotes to give input, as shown in the example). Use STL based stack.
3. Write a program which takes an expression in infix notation as a command line argument and converts it to postfix notation. For example, if the program is `infix2postfix.exe`, running `infix2postfix.exe "11 - ( 2 + 43 )"` should display `11 2 43 + -`. The operands and operators should be separated by spaces. Operands may consist of 1 or more digits. The input should be treated as a single command line argument (use quotes to give input, as shown in the example). Use STL based stack.
4. Implement an array based double ended queue ADT. The elements can be added and removed from both ends of the queue. For example, consider the following queue:

5	7	10	9

Front Rear

push_front(6) will enque an element at front.

6	5	7	10	9

Front Rear

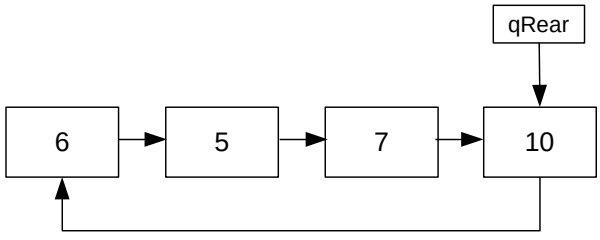
pop_rear() will remove the element from the rear (9 in this example).

6	5	7	10

Front **Rear**

In addition to standard queue ADT functions, these functions should be added: *push_front, push_rear, pop_front, pop_rear, front, rear, operator=, copy constructor*. All functions except operator= and copy constructor should be of O(1). No space should be wasted in the array (may be except for one element).

5. Implement a linked structures based circular queue ADT. Only keep a single pointer to the rear element. To access the front element, use the rear pointer. An example of a circular queue is shown below.



6. Implement a queue system for a hospital. The program should support multiple queues (for example, one for medical checkup, another for lab tests, another for something else, and so on). The number of queues will be given at runtime (during execution of the program). Queue should contain basic information about the visitors/patients. The user-interface should enable the user to add and remove persons from any queue. The user-interface should be intuitive and easy to use. The program should use the STL based queue.

Instructions:

- Start early.
- Before submission, remove all the debugging and temporary files (in visual studio select menu *Build → Clean Solution*). Only submit the .cpp and .h files (no visual studio or other files).
- Select .cpp and .h files and compress them using your full registration number and name, (e.g., `04071512007-Ali-Ahmad.zip`).
- Submit via email (rabbasi@qau.edu.pk) within due time (no extensions).
- The source code should be properly indented.
- Properly comment your source code.
- Try to avoid using conio.h, as it is not part of standard C++. Don't use clear screen function. Don't use getch function, instead use the standard getchar function (if required).
- Following these instructions carries marks.
- Any genuine efforts in each part, would result in at least 50% marks (in that part). Make sure you put your best efforts to solve every part. Each part carries its own marks. Different parts are highlighted in the assignment description (see the bold words).
- You are getting 50% marks for any genuine efforts in all the parts to encourage you to learn, even if your program does not compile and is full of bugs). Therefore, please do not plagiarize! Plagiarism includes taking help in any form including but not limited to code, concept or idea for the solution, algorithm, and pseudocode. Taking help from any source including but not limited to classmates, seniors, or internet is prohibited. In case your code is plagiarized, you'll get -50% absolute marks of the whole assignment. For example, if the assignment is of 50 marks, you will get -25 marks. **Even a single plagiarized statement will count as plagiarism for the whole assignment**. Plagiarism in two assignments would result in getting failed in the course.