Programming Assignment 1:
Stack and Queue Practice

Due date: 11:55 pm on Feb. 14, 2017 (Wednesday)

Submit to ReggieNet. (No email submission will be accepted.)

Description:
You are to write two C++ programs in a group of three people. The first will be practice with stacks; the second will be practice with queues. Both will also involve **linked lists**.

Stack program:
Develop a stack class using a linked list representation. The data for each node will contain a character and two integers.

You're going to use your stack class to provide unlimited undo and redo capabilities for a simple text calculator. The calculator will be integer-only. The user will enter a command possibly followed by a number. The calculator will perform the appropriate operation and report the current value of the calculation. The possible commands will be +, -, *, /, %, C, U, R and Q. The arithmetic operations will be followed by an integer (with or without a space between command and number) and represent the appropriate integer calculation. C means clear. It will set the current value to 0. U is undo. R is redo. Q is quit. Use ">" to prompt for the next command. If there's nothing to undo or redo, report that and then the unchanged current value.

At the beginning, provide a brief set of instructions to the user.

Note that you will need two different stacks to accomplish the task.

Sample partial program runs (user input in bold):

0
> **+300**
300
> **-500**
-200
>/ **100**
-2
>**R**
No operations to redo
-2
>**U**
-200
>**U**
300
>**R**
-200
>**C**
0
>**U**
-200
>+ **3**
-197
>**R**
No operations to redo
-197
>**Q**

Goodbye

0
> **+300**
300
> **-500**
-200
>/ **100**
-2
>**R**
No operations to redo
-2
>**U**
-200
>**U**
300
>**R**
-200
>**R**
-2
>**C**
0
>**U**
-2
>+ **3**
+1
>**R**
No operations to redo
+1
>**Q**

Goodbye

Queue program:
Develop a queue class using a linked list representation and integer data and use it in a program to simulate a check-out line at a supermarket. Customers arrive in random integer intervals of 1 to x minutes. Also, each customer is serviced in random integer intervals of 1 to x minutes. Request a value for x at the beginning of the run. Run the supermarket simulation for a 12-hour day (720 minutes) using the following algorithm:

1. Choose a random integer between 1 and x to determine the minute at which the first customer arrives.
2. At the first customer's arrival time:
   Print an arrival message;
   Determine customer's service time (random integer from 1 to x);
   Begin servicing the customer;
   Schedule the arrival time of the next customer (random integer 1 to x added to the current time)
3. For each subsequent minute of the day:
   If the next customer arrives,
      Print an arrival message;
      Enqueue the customer;
      Schedule the arrival time of the next customer;
   If service was completed for the last customer:
      Print a departure message;
      Dequeue the next customer to be serviced;
      Determine customer's service completion time;

The arrival and departure messages will look like:

Customer xx arrived at yy.
Customer xx left at yy.

Where xx represents which customer, counting from 1 (not zero), and yy is the current time (in the range 1-720).

At the end of the simulation, your program must print out the maximum number of customers in the queue at any one time, and the longest wait any one customer experienced. Note that the number of customers in the queue does not include the one being waited on, and the wait does not include service time.

Don't make up your own algorithm: use this one. Don't use a queue of customers, just ints, that's all you need if you do it as specified. Note that the algorithm does not specifically deal with every situation, however. Do complete the details so that it works fully.

General program requirements:
Your stack and queue classes must be good ADT implementations. Do not allow the specifics of the application to impact the design and implementation of those classes. Make sure that your classes use good dynamic memory management (so they must have correct destructors, copy constructors and assignment operators).

Submit the following items

(1) Group submission: (a) your source files from the School's linux machines using ReggieNet. Note that you should have at least three files (one .h and 2 .cpp) for each program. Name your files with clearly meaningful names (no main.cpp, please). Since you can attach no more than 5 files to an assignment, you will need to zip or tar your files. Please submit two files, one named either stack.zip or stack.tar, containing your stack program, the other named either queue.zip or queue.tar, containing your queue program. Include only the .h and .cpp files and a makefile if you have one, no folders, no backup files, no executables, nothing extra.

(2) Individual submission: Peer evaluations by each member.

IT 279 Program Grading Standards

An A program:
- Will follow principles of good structured and object-oriented programming
- Will be well-commented
- Will work perfectly
- Will have a stack or queue class that is a good implementation of the abstract data type
- Will have correct dynamic memory management
- Will be correctly formatted
- Will be correctly submitted

A B program:
- Will follow principles of good structured and object-oriented programming
- Will have a stack or queue class that is a good implementation of the abstract data type
- Will have correct dynamic memory management
- Will be reasonably well-commented
- Will work well
- May have one minor error in execution
  - or may have flaws in formatting and/or commenting
  - or may be incorrectly submitted

A C program:
- Will basically work
- May deviate somewhat from good structured programming
  - or may have a stack or queue class that is not a good ADT implementation
  - or may have multiple minor execution errors
  - or may have one serious execution error
  - or may be uncommented

A D program:
- Will run
- May use poor programming style
  - or may have multiple serious errors
  - or may have several problems of different kinds

An F program:
- Fails to compile
  - or fails to run
  - or fundamentally fails to fulfill the assignment
  - or has many different problems

IT 279

Programming Assignment 1: Peer Evaluations of Team Members

Every team member evaluates himself/herself and each fellow team member. Peer evaluations could affect your grade; thus, it is in your best interest to ensure that these evaluations are conducted and submitted to ReggieNet by **the designated date and time.**
Individual scores on each criterion should range from 1 *(very poor)* to 5 *(exceptionally outstanding).* Generally speaking, 1's should be as rare as 5's, given that few team members are outstandingly poor or outstandingly excellent. Provide a score for each team member (including yourself!) on each criterion, and provide comments to justify any very low or very high scores. Also compute an average score for each team member.
Each team member must complete a peer evaluation form to the instructor. *These will be treated as confidential.* The peer reviews will be used to identify whether an individual or group collaboration was even, to apportion the group grade. Please upload individual team evaluations to **Reggie**Net by the deadline of Programming Assignment 1

(Here, roles mean the roles in POGIL.): manager, recorder, and strategy analyst

	Your name and role	**Member1's name and his/her role**	**Member2's name and his/her role**
Group leadership or ability to plan project and help keep team on track			
Attitude toward the discussion and team members			
Regular attendance at team meetings (or collaborating via email) and being prepared for meetings/collaboration			
Willingness to accept tasks			
Ability and willingness to constructively criticize, synthesize, or edit and assemble group work			
Ability to understand what needs to be done and to interpret this for the group			
Quality of individual work towards assigned task			
Timeliness of individual work towards assigned task			
Percentage of effort in completing the team discussion			
Contribution over/above the call of duty, such as willingness to help others, or taking on extra work			
Average			

comments: