CIS 291 – Data Structures in C++ - Spring 2005
Week 3 Lab Exercise

Week 3 Lab Exercise due: Tuesday, February 1st, END of lab

Purpose: practice related to stacks and queues

1. INDIVIDUAL-answer, TEAM-verification exercise:

On a sheet of paper with your name on it, **individually** answer all of the questions below. Once you have written out your initial answers **by yourself**, **then** compare them with at least one other classmate's answers. If they differ, discuss why until you both agree on the answer (and change the answer on your paper accordingly if appropriate). Write down the name(s) of all those you conferred with on your paper. When all in your "group" have agreed on answers, then put on of the group's names on the **next:** list.

Consider the following sequence of stack and queue operations; I want you to **hand-execute** them, to see if you are comfortable with how stacks and queues work. (Assume that, as this begins, aStack is an empty stack of integers, and aQueue is an empty queue of integers.) Their implementation is unimportant, however DO note that they do correspond to the **stack** and **queue** pseudo-UML's given out in lecture (and now available from the public course web page).

(Follow the directions carefully; the point here is to make sure you are comfortable with how a stack and queue "behave".)

```
stack<int>      aStack;
queue<int>      aQueue;
int             looky1, looky2, looky3;

aStack.push(2);
aStack.push(4);
aStack.push(6);
aStack.push(8);

/* POINT A: write POINT A, and draw aStack's CONTENTS at this point, clearly
   labeling the stack's TOP */

aStack.pop( );
aStack.push(10);
aStack.pop( );

/* POINT B: write POINT B, and draw aStack's CONTENTS at this point, clearly
   labeling the stack's TOP */

looky1 = aStack.get_top( );
aStack.pop( );

looky2 = aStack.get_top( );

looky3 = aStack.pop( );

/* POINT C: write out the output of the following statements at this point: */

cout << "POINT C" << endl;
cout << "looky1: " << looky1 << endl;
cout << "looky2: " << looky2 << endl;
cout << "looky3: " << looky3 << endl;

aQueue.enqueue(2);
aQueue.enqueue(4);
aQueue.enqueue(6);
aQueue.enqueue(8);

/* POINT D: write POINT D, and draw aQueue's CONTENTS at this point, clearly
   labeling the queue's FRONT and REAR */

aQueue.dequeue( );
aQueue.enqueue(10);
aQueue.dequeue( );

/* POINT E: write POINT E, and draw aQueue's CONTENTS at this point, clearly
   labeling the queue's FRONT and REAR */

looky1 = aQueue.get_front( );

looky2 = aQueue.get_front( );
aQueue.dequeue( );

looky3 = aQueue.dequeue( );

/* POINT F: write out the output of the following statements at this point: */
cout << "POINT F" << endl;
cout << "looky1: " << looky1 << endl;
cout << "looky2: " << looky2 << endl;
cout << "looky3: " << looky3 << endl;
```

Your responses must be checked before lab is over.

2. You are exercising a **stack** implementation in HW #2; you will exercise a **queue** implementation here.

From the course web page, copy over the following files: **queue.h**, **queue.template**

(If you are copying into pico, beware its "helpful" tendency to add a new line to long lines, which is unfortunate when the long line is a single-line comment or a string literal...)

Using the posted template for a testing main **for a class** as your basis, write a **test_queue.cpp** tester main function, that also meets the following additional requirements:

* appropriately inserts at least five values into an example queue which is a queue of **char**'s;

* write a statement that prints to the screen what SHOULD be in the queue at this point (if you consider the queue's front to be on the **left** and the queue's rear to be on the **right**).

* write a while loop that continues while the queue is not empty, repeatedly dequeuing and printing the front value on the queue (so, yes, this is a destructive loop...)

* (you may add any additional stack and queue playing around that you wish, but make sure that you at least follow the above specifications.)

When you are satisfied with your program, run:

```
test_queue > 291lab03_2_out
```

... and put your name on the Next: list, to have your test_queue.cpp and 291lab03_2_out checked over.

To receive credit for this lab exercise, the above must be completed by the end of the lab period.