

Due 1:30PM Wednesday, September 15

Name _____

Student ID _____

Instructions Follow instructions *carefully*, failure to do so will result in points being deducted. It is highly recommended that you write your homework using \LaTeX or Word. Staple this cover page to the front of your assignment for easier grading. Be sure to submit a hardcopy of your problem answers and your program analysis in class. Clearly label each problem and submit the answers in order. Put your program analysis to the end. In addition, submit your homework including all the program files via the webhandin (<http://www.cse.unl.edu/~cse310/handin>). Late homework policy specified in course syllabus *will be strictly followed*. Be sure to show *sufficient work* to justify your answer(s). Numerical answers with no explanation will NOT be granted full points. If you are asked to prove something, you must give as formal, rigorous, and complete proof as possible. You are to work individually, and all work should be your own. The CSE academic dishonesty policy is in effect (http://cse.unl.edu/undergrads/academic_integrity.php).

Problem	Points	Score
1	10	
2	10	
3	5	
4	10	
5	15	
Program		
Correctness	30	
Style/Doc	10	
Analysis	10	
Total	100	

Problems

1. Use *mathematical induction* to prove that

$$1 \times 2 \times 3 + 2 \times 3 \times 4 + \dots + n(n+1)(n+2) = n(n+1)(n+2)(n+3)/4.$$

2. How many strings of three decimal digits

- (a) do not contain the same digit three times?
- (b) begin with an odd digit?
- (c) have exactly two digits that are 4s?

3. There are 38 different time periods during which classes at a university can be scheduled. If there are 677 different classes, how many different rooms will be needed?

4. A computer company is selling 90 PC's with the following optional features: (i) Windows XP (ii) CD-RW (iii) 27-inch monitors. Among these PC,

- 72 have at least one of the options (i) or (ii).
- 18 have both options (i) and (ii).
- 45 have option (iii).
- 36 have at least two of the options (i), (ii) or (iii).
- 72 *do not* have at least one of the options (i) or (ii) or (iii).

(a) How many PC's have options (i) and (ii) but not (iii)?

(b) How many PC's have at least one of the options (i), (ii) or (iii)?

5. Order the following functions according to their order of growth (from the lowest to the highest).
 $(n-2)!$, $5lg(n + 100)^{10}$, 2^{2n} , $0.001n^4 + 3n^3 + 1$, ln^2n , $\sqrt[3]{n}$, 3^n .

Program

The goal of the program will be to implement a simulation of a *priority queue* data structure using two stack data structures. You may not implement the priority queue directly. Recall that a priority queue, rather than being FIFO dequeues the element with the highest priority. You must first figure out how to implement the **enqueue** and **dequeue** operations of the queue in terms of the **push** and **pop** operations of the two stacks. There are clever ways of doing this, but efficiency will not be an issue, only correctness. The queue will only hold data of the `int` type but should have no maximum size. Further, the maximum priority will be the smallest valued integer. You should define and implement all data structures yourself, *do not use predefined libraries*.

Your program must be written in C++, execute correctly on the CSE unix environment, and compile using the `g++` compiler using a `makefile`. Specifically, you will hand in the following files:

- `makefile` – the makefile to compile your program into an executable called `mypriorityqueue`
- `main.cpp` – the main program that will parse an input file (see below)
- `stack.h` – the header file for the `stack` class.
- `stack.cpp` – the implementation of your `stack` class.
- `Pqueue.h` – the header file for the `Pqueue` class.
- `Pqueue.cpp` – the implementation file for your `Pqueue` class.

Be sure to follow the file naming conventions listed above. In addition, you must follow the function naming conventions listed below. These functions are a minimum of what you must include in your class, you may define other member functions as you wish.

Your `stack` class will have the following member functions (note that a default constructor is also required).

- `void push(int x)` – the push function.
- `int pop()` – the pop function.
- `bool is_empty` – a boolean test to see if the stack is empty.

If a `pop` operation is performed on an empty stack, *no error message should be echoed*, instead it should have no effect at all.

Your `Pqueue` class will have the following member functions, again a default constructor is implied.

- `void enqueue(int x)` – enqueue the integer `x`. If `x` is already in the queue, it should have no effect, thus the priority queue should have *all unique elements*

- `int dequeue()` – dequeue the element with the highest priority (the smallest valued integer).

The `main.cpp` program should take input from a file via the command line (example: `prompt:>mypriorityqueue inputfile`) with the following structure in the input file (you may assume that all input file will be well structured so you don't have to error check the input).

- `enq x` which should perform the operation `enqueue(x)`
- `deq` which should dequeue the first element in the queue. If the queue is empty then the command is ignored (do not echo an error).
- `print` which should output (to the standard output) the current contents of the queue. Since its a priority queue, the order does not matter but the entire contents should be printed to a single line delimited by single spaces. If the queue is empty then simply output `empty`. Note, this command should not affect the queue contents, simply print them.

An example of an input file is as follows:

```
enq 5
enq 10
deq
enq 3
print
enq 4
deq
enq 50
print
deq
print
deq
deq
print
```

Proper output should look (something) like:

```
10 3
4 10 50
10 50
empty
```

Points will be awarded based on the following categories:

- **Correctness** – Your code should execute as described above, naming conventions should be followed. You must include a makefile that will compile your code into an executable.
- **Style/Documentation** – Your code should be readable, properly indented and spaced with sufficient comments to explain. Generally good Object Oriented style should be followed including separation of header/implementation files, appropriate naming of methods and variables, etc.
- **Analysis** – Write a **brief** analysis of how your program works. In particular you should describe how your priority queue was implemented using the two stacks. Try to discuss how much more complicated the operations are over the usual implementations of priority queues. Discuss any pitfalls that you faced and any other relevant issues.