C243/A594 **Programming Assignment #4** Spring 2025

Priority Queues

General: In this assignment you will be implementing your own collection class called a **priority queue**, which stores its elements in an order determined by an integer ***priority value***. Much like the NCAA basketball tournament seedings, lower values denote a higher priority, or one that is more "urgent". A priority queue can be implemented in various ways including using an array, which is what is required for this assignment. In addition to creating your collection class, other classes are required to provide functionality that allows a user to perform the fundamental operations of adding elements to and removing elements from a priority queue. Other actions invoked on a priority queue will be implemented.

Objectives: This assignment is designed to give you the opportunity to write your own collection class along with a class defining the data items. Additionally, you will be creating a class to house the **main** function and a "helper method" that handles the user interface tasks.

Specifics:

1) The basic idea of this assignment is to allow the client (user) to create tasks that need to be accomplished (e.g. mowing lawn, doing laundry, going shopping, etc.) and prioritizing them. New tasks can be added and the priority queue will arrange them with the most urgent being at the root. Tasks can be removed from the queue when they are undertaken.

2) The operations that are available to the **client** are given in Table 1. You should note that these methods are similar to ones used for a general Queue class. Differences are that the **enqueue** method takes an extra argument denoting the priority of the task to be accomplished. Additionally, there is a new method that returns the priority associated with the most urgent item in the queue.

Table 1: Methods available to the client in the PriorityQueue class.

enqueue (*task, priority*)	Adds *task* at the appropriate position in the queue as determined by the integer *priority*. Lower integer values denote a greater sense of urgency.
string dequeue ( )	Removes the item at the root of the queue and returns the *task* to the user. Calling this function on an empty queue causes a runtime error and **must be avoided**.
string peek ( )	Returns the value of the most urgent item in the queue without removing it. Calling this function on an empty queue causes a runtime error and **must be avoided**.
int peekPriority ( )	Returns the priority of the most urgent item in the queue without removing it. Calling this function on an empty queue causes a runtime error and **must be avoided**.
clear ( )	Removes all elements from the queue.
int size ( )	Returns the number of elements currently in the queue.
bool isEmpty ( )	Returns **true** if the queue is empty.

3) You are to write a container class, name it **priorityQueue,** that will contain the code for the methods shown in Table 1. You are also to write a class to house **main**, name it **pqTest**, that will also have a method to ***print*** the menu and ***return*** the user's selection. The format for this method is **int printMenu()** and ***must*** implement the following:

Type a choice from the menu:

0) EXIT the program.
1) Enqueue a task.
2) Dequeue a task.
3) Peek at the most urgent task.
4) Peek at the priority of the most urgent task.
5) Clear the queue of all elements.
6) Return the number of tasks in the queue.
7) Determine if the queue is empty.

4) The priority queue is to be implemented by using an **array** of **Job** objects. Recall that to do this you need to first declare the array and second, declare the individual objects. You need to create this class using the following attributes and appropriate accessor and mutator methods: Question: Do you think that you need to also write a **constructor**?

```
class Job{
        private int priority;
        private string taskName;

        int getPriority();
        string getTaskName();
        void setPriority(int val);
        void setTaskName(string tname);   };
```

5) When invoking the program, the first task is to print the menu and receive the selection. Then, the selection needs to be carried out. Note: When executing options that could possibly cause a runtime error, be sure that you call the appropriate function that will indicate whether the action should be carried out. If it cannot be executed an appropriate message should be output to the terminal screen.

6) Priority Queue implementation – use an array as the underlying representation. You need to write the required methods in the context of the array model. With the **enqueue** and **dequeue** functions the **upheap** and **downheap** operations are required. Write your code so that it ***very closely*** matches the code given in class and posted on the course website. Solutions that significantly deviate from the code provided will not be accepted. Define and use a constant, CAPACITY, that is set to 10. This is used to denote the maximum capacity of the priority queue. Hence, there can be no more than CAPACITY items in the priority queue.

7) Design a user-friendly and informative user interface (UI). Prompts must clearly ask for data from the user in the format required. Outputs must be prefaced indicating what is being sent to the terminal screen.

8) Make sure that you thoroughly test your program. As a reminder, your program must compile to receive any credit. Further, none of the menu selections can result in a segmentation fault/abnormal program termination/infinite loop. Finally, make sure that your solution executes properly on the computer science lab machines.

Submission Information:

1) Submit your program by 11:59pm on Tuesday April 22nd to the appropriate Canvas Assignments folder. On Wednesday, April 23rd at the start of class, submit a hard-copy of the program, properly formatted.

2) Create and submit a one-two paragraph document (name it "Statement of Functionality") that discusses the functionality of your program. Submit the electronic version of this to the Canvas folder by Tuesday, April 22nd at 11:59pm with the printout being submitted at the start of class on Wednesday, April 23rd. Be honest and complete in the assessment of your work. Note: It is **imperative** that you are honest about any program shortcomings. If you state that your program functions properly and during testing it is found to have errors, the work will incur a substantial additional penalty. In the "Statement of Functionality" be sure to include a paragraph or more discussing what you did to test your program - be complete.

Sample Output

```
> enqueue mow lawn        3
> enqueue haircut         1
> enqueue do laundry      4
> enqueue take nap        2
> dequeue
  Task being removed is:  haircut
> peek
  Highest priority task: take nap
> peekPriority
  The Highest priority task has priority: 2
> isEmpty
  Task list is not empty.
> clear
   Task list has been cleared.
> isEmpty
  Task list is empty.
> peek
 There are no tasks in the list.
```