**Assignment 2: Implementing a templatized Queues ADT (QueueADT).**

Part A: The ADT is called **QueueADT**

//Queue has exactly three integer variables: rear, front and end. Don’t use count.

Required Functions:

1. bool Enque (T a)
2. bool Deque (T &a)
3. bool IsEmpty()
4. bool IsFull()
5. QueueADT() //default constructor. Creates a queue of default size say 10
6. QueueADT(int size) constructor. Creates a queue of size = size

You will also need a destructor.

PartB: The main/driver program

1. Declare an integer and character QueueADT objects using the overloaded constructor.
2. Then the following code should execute without errors and WITHOUT ANY CHANGES. I will copy paste this exact code in main and if there is an error of any sort, your entire code will be considered incorrect and incomplete.

cout<<obj1.Enque(newval);

obj1. Enque (1);

obj1. Enque (2);

obj2. Enque ('3');

obj1. Enque (5);

if(!obj1.IsFull())

obj1. Enque (4);

for(int i=0; i<6; i++)

{

obj1.Deque(newval);

cout<<newval;

}

Part C: Application of QueueADT: **Simulating priority queues**.

1. Create a program structure. It contains:

* float ProgramAge (representing creationTime),
* integer programNumber and
* integer priority\_level (0 = LOW, 1 = MID and 2= LOW)**.**

ProgramNumber is a just like a counter. Program age can have values 0<ProgramAge<15

1. Create three queues of type program: QueueADT<program> low, high and mid.
2. Create a function **program getNextProgram()**, that creates a program, randomly assigns values to its programAge and priority\_level (0,1 or 2) and returns the program. It also assigns value to programNumber. let programNumber = 0 for first program, programNumber=1 for second and so on.
3. Get a program from **getNextProgram**(). Based on the priority of the program, select the queue for the program.
4. Once the queue is decided, insert the program into the queue depending on its ProgramAge. The program with the highest ProgramAge should be in front of the queue.
5. Also, write the ProgramAge, programNumer and creationTime of this program in a text file named “**progseq.txt**”. This file has the order in which the program were CREATED.
6. Repeat steps 4-6, exactly 6 times.
7. And finally, once all the programs are in their respective queues, start processing the programs by getting a program from the queue one at a time moving from high to low. Pass each selected program to a function **ProcessThis(program p)**, that is a dummy function and only prints the contents of the program in a file called “**config.txt**”: This file has the order in which the program were PROCESSED.

**Format of file:** file should look something like:

High:

ProgramNumber = 2, ProgramAge=3.5

ProgramNumber = 4, ProgramAge=2

…

Mid:

ProgramNumber = 0, ProgramAge=13.2

…

Low:

ProgramNumber = 3, ProgramAge=4.0

ProgramNumber = 1, ProgramAge=3.0

…

**ALERT! THIS IS AN APPLICATION OF QUEUE. DO NOT MODIFY THE QUEUEADT FOR THIS PROGRAM.**