```
                                        /** Queue.h by Robert Szkutak */

#ifndef QUEUE_H
#define QUEUE_H

#include <iostream>

#define MAX_ARRAY_SIZE 10//Maximum size of the Queue

typedef myQueueType int;//The type of elements the Queue will hold

class Queue
{
        private :
                myQueueType myArray[MAX_ARRAY_SIZE];//Array of elements in the Queue
                int size;//Size of the Queue
        public :
                Queue();//Constructor
                ~Queue();//Destructor
                myQueueType dequeue();//Removes element from Queue
                void enqueue(myQueueType var);//Adds element to Queue
                void empty();//Empties the Queue
                bool isFull();//Returns true if the Queue is full
                bool isEmpty();//Returns true if the Queue is empty
                void printQueue();//Uses STL to output the contents of Queue
};

#endif
```

```
/** Queue.cpp by Robert Szkutak */

#include “Queue.h”

/**
```

```
   The constructor for the Queue class
 */
Queue::Queue(){empty();}

/**
   The destructor for the Queue class
 */
Queue::~Queue(){}

/**
   Dequeues an element in the Queue
   @return the element Dequeued
 */
myQueueType Queue::dequeue()
{
	if(!isEmpty())
	{
		myQueueType ret = myArray[0];
		for(int i = 0; i < MAX_ARRAY_SIZE-1; i++)
			myArray[i] = myArray[i+1];
		size--;
		return ret;
	}
	return -1;//Erorr
}

/**
   Enqueues an element in the Queue
   @param the element to Enqueue
 */
void Queue::enqueue(myQueueType var)
{
	if(!isFull())
	{
		size++;
		myArray[size] = var;
	}
}

/**
   Empties the Queue
 */
void Queue::empty()
{
	size = -1;
}

/**
    Tests to see if the Queue is empty
```

```
    @return true if the Queue is empty, false if it is not
 */
bool Queue::isEmpty()
{
        if(size <= -1)
                return true;
        return false;
}

/**
    Tests to see if the Queue is full
    @return true if the Queue is full, false if it is not
 */
bool Queue::isFull()
{
        if(size >= MAX_ARRAY_SIZE-1)
                return true;
        return false;
}

/**
   Outputs the contents of the Queue
 */
void Queue::printQueue()
{
        char pause = 0;

        for(int i = 0; i < MAX_ARRAY_SIZE-1; i++)
                std::cout << i + " " myArray[i]; + "\n";
        std::cout << "\nPress ENTER to continue\n\n";
        std::cin >> pause;
}
/** main.cpp by Robert Szkutak */

#include <ctime>//Included for random number generation
#include "Queue.h"

void Test1(Queue queue);
void Test2(Queue queue);
void Test3(Queue queue);

int main()
{
        Queue queue;

        srand(time(0));//Seeds the random number generator

        Test1(queue);
```

```
	queue.empty();
	Test2(queue);
	queue.empty();
	Test3(queue);

	return 0;
}

/**
     Tests Enqueuing several elements and Dequeuing a couple elements in the Queue
    @param the Queue to be tested
 */
void Test1(Queue queue)
{
	for(int i = 0; i < 5; i++)
		queue.enque(rand() % 100 + 1);
	queue.dequeue();
	queue.dequeue();
	queue.printQueue();
}

/**
    Tests Enqueuing too many elements in the Queue
    @param the Queue to be tested
 */
void Test2(Queue queue)
{
	for(int i = 0; i < MAX_ARRAY_SIZE*3; i++)
		queue.enque(rand() % 100 + 1);
	queue.printQueue();
}

/**
    Tests Dequeuing too many elements from the Queue
    @param the Queue to be tested
 */
void Test3(Queue queue)
{
	for(int i = 0; i < MAX_ARRAY_SIZE*3; i++)
		queue.dequeue();
	queue.printQueue();
}
```