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

#ifndef HEAP_H
#define HEAP_H

#include <iostream>

#define MAX_ARRAY_SIZE 10//Maximum size of the Heap

typedef myHeapType int;//The type of elements the Heap will hold

class Heap
{

	private :
		myQueueType myArray[MAX_ARRAY_SIZE];//Array of elements in the Heap
		int size;//Size of the Heap
	public :
		Heap();//Constructor
		~Heap();//Destructor
		myHeapType pop();//Removes an element from the Heap
		void push(myHeapType var);//Adds an element to the Heap
		void empty();//Empties the Heap
		bool isFull();//Returns true if the Heap is full
		bool isEmpty();//Returns true if the Heap is empty
		void printHeap();//Uses STL to output the contents of Heap
};

#endif
```

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

#include “Heap.h”

/**
  The constructor for the Heap class
*/
```

```
Heap::Heap(){empty();}

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

/**
  Removes an element from the Heap
  @return the element popped off the Heap
*/
myHeapType Heap::pop()
{
	if(!isEmpty())
	{
		myHeapType ret = myArray[0];
		for(int i = 0; i < MAX_ARRAY_SIZE-1; i++)
			myArray[i] = myArray[i+1];
		size--;
		return ret;
	}
	return -1;//Erorr
}

/**
  Adds an element to the Heap
  @param the element to add to the Heap
*/
void Heap::push(myHeapType var)
{
	int buffer;

	if(!isFull())
	{
		size++;
		for(int i = 0; i <= size; i++)
		{
			if(MyArray[i] < var)
			{
				buffer = MyArray[i];
				MyArray[i] = var;
				for(int j = i; j <= size; j++)
				{
					var = buffer;
					buffer = MyArray[i];
					MyArray[i] = var;
				}
				break;
```

```
                                }
                        }
                }
}

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

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

/**
  Outputs the contents of the Heap
*/
void Heap::printHeap()
{
        char pause = 0;
        if(isEmpty())
        {
                std::cout << "The Heap is empty\n\n";
                return;
        }
        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 "Heap.h"

void Test1(Heap heap);
void Test2(Heap heap);
void Test3(Heap heap);

int main()
{
	Heap heap;

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

	Test1(heap);
	heap.empty();
	Test2(heap);
	heap.empty();
	Test3(heap);

	return 0;
}

/**
    Tests adding pushing elements and popping a couple elements to and from the Heap
   @param the Heap to be tested
*/
```

```
void Test1(Heap heap)
{
	for(int i = 0; i < 5; i++)
		heap.push(rand() % 100 + 1);
	heap.pop();
	heap.pop();
	heap.printHeap();
}

/**
   Tests pushing too many elements to the Heap
   @param the Heap to be tested
*/
void Test2(Heap heap)
{
	for(int i = 0; i < MAX_ARRAY_SIZE*3; i++)
		heap.push(rand() % 100 + 1);
	heap.printHeap();
}

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

Program Output:

```
63
78
94

Press ENTER to continue

16
33
41
58
62
66
79
83
87
91

Press ENTER to continue

The Heap is empty

Press ENTER to continue
```