**Lab Description:**

\*\*Description in README file

\*\*Not on lab write up

**Main.cpp:**

#include <iostream> // cout, endl

#include <stdio.h> // printf

#include "HeapQ.cpp" // HeapQ class and HeapObj struct

#include <string> // std::string class/data type

int main() {

std::cout << "Testing HeapQ of integers:" << std::endl;

HeapQ<int> intQ;

std::cout << "Quick test of empty print:" << std::endl;

intQ.print();

std::cout << "Testing Enqueue Function:" << std::endl;

intQ.enqueue(25, 10);

intQ.enqueue(23, 55);

intQ.enqueue(25, 100);

intQ.print();

std::cout << "Testing Copy Constructor: (should match above output)" << std::endl;

HeapQ<int> intQ2(intQ);

intQ2.print();

std::cout << "Testing Overloaded Assignment: (should match above)" << std::endl;

HeapQ<int> intQ3 = intQ2;

intQ3.print();

std::cout << "Preventing self assignment" << std::endl;

intQ3 = intQ3;

std::cout << "Testing Enqueueing over and resizing array:" << std::endl;

intQ.enqueue(87, 40);

intQ.enqueue(67,90);

intQ.enqueue(14, 125);

intQ.enqueue(-40, 500);

intQ.enqueue(222, 32);

intQ.enqueue(40, 88);

intQ.enqueue(4, 2);

intQ.print();

std::cout << "Testing peek Function:" << std::endl;

HeapQ<int> intQ4;

intQ4.enqueue(20, 1);

intQ4.enqueue(25, 90);

intQ4.enqueue(10, 3);

std::cout << "Working queue..." << std::endl;

intQ4.print();

std::cout << "Peeking off the top..." << std::endl;

intQ4.peek();

std::cout << "Testing Dequeue Function:" << std::endl;

std::cout << "Working queue" << std::endl;

intQ.print();

int returned = intQ.dequeue();

std::cout << "Number that was returned..." << std::endl;

std::cout << returned << std::endl;

std::cout << "Resulting queue after clean up" << std::endl;

intQ.print();

std::cout << "Running some more dequeues and then enqueueing..." << std::endl;

int returned2 = intQ.dequeue();

int returned3 = intQ.dequeue();

int returned4 = intQ.dequeue();

std::cout << "Numbers returned from three dequeues..." << std::endl;

std::cout << returned2 << ", " << returned3 << ", " << returned4 << std::endl;

std::cout << "Current queue..." << std::endl;

intQ.print();

std::cout << "Enqueueing a couple numbers..." << std::endl;

intQ.enqueue(5, 1);

intQ.enqueue(23, 62);

std::cout << "Resulting queue..." << std::endl;

intQ.print();

std::cout << "All queue related functions (except print() for testing purposes)" <<

" will throw an error (string) and will display an error message if" <<

" caught properly" << std::endl;

std::cout << "Testing HeapQ of doubles:" << std::endl;

HeapQ<double> doubQ;

std::cout << "Quick test of empty print:" << std::endl;

doubQ.print();

std::cout << "Testing Enqueue Function:" << std::endl;

doubQ.enqueue(4.555, 7);

doubQ.enqueue(78.3, 67);

doubQ.enqueue(67.111, 78);

doubQ.print();

std::cout << "Testing Copy Constructor: (should match above output)" << std::endl;

HeapQ<double> doubQ2(doubQ);

doubQ2.print();

std::cout << "Testing Overloaded Assignment: (should match above)" << std::endl;

HeapQ<double> doubQ3 = doubQ2;

doubQ3.print();

std::cout << "Preventing self assignment" << std::endl;

doubQ3 = doubQ3;

std::cout << "Testing Enqueueing over and resizing array:" << std::endl;

doubQ.enqueue(78.456, 9);

doubQ.enqueue(98.27529, 90);

doubQ.enqueue(879.45, 1000);

doubQ.enqueue(729, 78);

doubQ.enqueue(89, 909);

doubQ.enqueue(67.5666, 900);

doubQ.enqueue(75.6545, 234);

doubQ.print();

std::cout << "Testing peek Function:" << std::endl;

HeapQ<double> doubQ4;

doubQ4.enqueue(54.4, 90);

doubQ4.enqueue(43.567, 80);

doubQ4.enqueue(78.54, 22);

std::cout << "Working queue..." << std::endl;

doubQ4.print();

std::cout << "Peeking off the top..." << std::endl;

doubQ4.peek();

std::cout << "Testing Dequeue Function:" << std::endl;

std::cout << "Working queue" << std::endl;

doubQ.print();

double ret = doubQ.dequeue();

std::cout << "Number that was returned..." << std::endl;

std::cout << ret << std::endl;

std::cout << "Resulting queue after clean up" << std::endl;

doubQ.print();

std::cout << "Testing HeapQ of strings" << std::endl;

HeapQ<std::string> strQ;

std::cout << "Quick test of empty print:" << std::endl;

strQ.print();

std::cout << "Testing Enqueue Function:" << std::endl;

strQ.enqueue("jarvis", 80);

strQ.enqueue("james", 90);

strQ.enqueue("Ryan", 100);

strQ.print();

std::cout << "Testing Copy Constructor: (should match above output)" << std::endl;

HeapQ<std::string> strQ2(strQ);

strQ2.print();

std::cout << "Testing Overloaded Assignment: (should match above)" << std::endl;

HeapQ<std::string> strQ3 = strQ2;

strQ3.print();

std::cout << "Preventing self assignment" << std::endl;

strQ3 = strQ3;

std::cout << "Testing Enqueueing over and resizing array:" << std::endl;

strQ.enqueue("ricky", 91);

strQ.enqueue("skylar", 30);

strQ.enqueue("jiminy", 9000);

strQ.enqueue("erika", 101);

strQ.enqueue("Gerald", 900);

strQ.enqueue("Hailey", 80);

strQ.enqueue("ariela", 566);

strQ.print();

std::cout << "Testing peek Function:" << std::endl;

HeapQ<std::string> strQ4;

strQ4.enqueue("armaya", 90);

strQ4.enqueue("jakob", 900);

strQ4.enqueue("harold", 55);

std::cout << "Working queue..." << std::endl;

strQ4.print();

std::cout << "Peeking off the top..." << std::endl;

strQ4.peek();

std::cout << "Testing Dequeue Function:" << std::endl;

std::cout << "Working queue" << std::endl;

strQ.print();

std::string ret1 = strQ.dequeue();

std::cout << "Value that was returned..." << std::endl;

std::cout << ret1 << std::endl;

std::cout << "Resulting queue after clean up" << std::endl;

strQ.print();

std::cout << "Quick note! All functions except print() will throw a string " <<

"error if out of bounds of something is not correct. Everything " <<

"else works in accordance with the rules of a priority queue." << std::endl;

return 0;

}

**Sample Output:**

Testing HeapQ of integers:

Quick test of empty print:

No items in the queue to print

Testing Enqueue Function:

Front of Queue

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

25

With Priority:

10

==========================

Back of Queue

Testing Copy Constructor: (should match above output)

Front of Queue

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

25

With Priority:

10

==========================

Back of Queue

Testing Overloaded Assignment: (should match above)

Front of Queue

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

25

With Priority:

10

==========================

Back of Queue

Preventing self assignment

Testing Enqueueing over and resizing array:

Front of Queue

==========================

Data:

-40

With Priority:

500

==========================

==========================

Data:

14

With Priority:

125

==========================

==========================

Data:

67

With Priority:

90

==========================

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

40

With Priority:

88

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

87

With Priority:

40

==========================

==========================

Data:

222

With Priority:

32

==========================

==========================

Data:

25

With Priority:

10

==========================

==========================

Data:

4

With Priority:

2

==========================

Back of Queue

Testing peek Function:

Working queue...

Front of Queue

==========================

Data:

25

With Priority:

90

==========================

==========================

Data:

10

With Priority:

3

==========================

==========================

Data:

20

With Priority:

1

==========================

Back of Queue

Peeking off the top...

HeapObj at front of the queue:

==========================

Data:

25

With Priority:

90

==========================

Testing Dequeue Function:

Working queue

Front of Queue

==========================

Data:

-40

With Priority:

500

==========================

==========================

Data:

14

With Priority:

125

==========================

==========================

Data:

67

With Priority:

90

==========================

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

40

With Priority:

88

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

87

With Priority:

40

==========================

==========================

Data:

222

With Priority:

32

==========================

==========================

Data:

25

With Priority:

10

==========================

==========================

Data:

4

With Priority:

2

==========================

Back of Queue

Number that was returned...

-40

Resulting queue after clean up

Front of Queue

==========================

Data:

14

With Priority:

125

==========================

==========================

Data:

25

With Priority:

100

==========================

==========================

Data:

67

With Priority:

90

==========================

==========================

Data:

222

With Priority:

32

==========================

==========================

Data:

40

With Priority:

88

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

87

With Priority:

40

==========================

==========================

Data:

4

With Priority:

2

==========================

==========================

Data:

25

With Priority:

10

==========================

Back of Queue

Running some more dequeues and then enqueueing...

Numbers returned from three dequeues...

14, 25, 67

Current queue...

Front of Queue

==========================

Data:

40

With Priority:

88

==========================

==========================

Data:

87

With Priority:

40

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

222

With Priority:

32

==========================

==========================

Data:

25

With Priority:

10

==========================

==========================

Data:

4

With Priority:

2

==========================

Back of Queue

Enqueueing a couple numbers...

Resulting queue...

Front of Queue

==========================

Data:

40

With Priority:

88

==========================

==========================

Data:

23

With Priority:

62

==========================

==========================

Data:

23

With Priority:

55

==========================

==========================

Data:

87

With Priority:

40

==========================

==========================

Data:

25

With Priority:

10

==========================

==========================

Data:

4

With Priority:

2

==========================

==========================

Data:

5

With Priority:

1

==========================

==========================

Data:

222

With Priority:

32

==========================

Back of Queue

All queue related functions (except print() for testing purposes) will throw an error (string) and will display an error message if caught properly

Testing HeapQ of doubles:

Quick test of empty print:

No items in the queue to print

Testing Enqueue Function:

Front of Queue

==========================

Data:

67.111

With Priority:

78

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

4.555

With Priority:

7

==========================

Back of Queue

Testing Copy Constructor: (should match above output)

Front of Queue

==========================

Data:

67.111

With Priority:

78

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

4.555

With Priority:

7

==========================

Back of Queue

Testing Overloaded Assignment: (should match above)

Front of Queue

==========================

Data:

67.111

With Priority:

78

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

4.555

With Priority:

7

==========================

Back of Queue

Preventing self assignment

Testing Enqueueing over and resizing array:

Front of Queue

==========================

Data:

879.45

With Priority:

1000

==========================

==========================

Data:

89

With Priority:

909

==========================

==========================

Data:

67.5666

With Priority:

900

==========================

==========================

Data:

98.2753

With Priority:

90

==========================

==========================

Data:

75.6545

With Priority:

234

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

78.456

With Priority:

9

==========================

==========================

Data:

729

With Priority:

78

==========================

==========================

Data:

4.555

With Priority:

7

==========================

==========================

Data:

67.111

With Priority:

78

==========================

Back of Queue

Testing peek Function:

Working queue...

Front of Queue

==========================

Data:

54.4

With Priority:

90

==========================

==========================

Data:

43.567

With Priority:

80

==========================

==========================

Data:

78.54

With Priority:

22

==========================

Back of Queue

Peeking off the top...

HeapObj at front of the queue:

==========================

Data:

54.4

With Priority:

90

==========================

Testing Dequeue Function:

Working queue

Front of Queue

==========================

Data:

879.45

With Priority:

1000

==========================

==========================

Data:

89

With Priority:

909

==========================

==========================

Data:

67.5666

With Priority:

900

==========================

==========================

Data:

98.2753

With Priority:

90

==========================

==========================

Data:

75.6545

With Priority:

234

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

78.456

With Priority:

9

==========================

==========================

Data:

729

With Priority:

78

==========================

==========================

Data:

4.555

With Priority:

7

==========================

==========================

Data:

67.111

With Priority:

78

==========================

Back of Queue

Number that was returned...

879.45

Resulting queue after clean up

Front of Queue

==========================

Data:

89

With Priority:

909

==========================

==========================

Data:

75.6545

With Priority:

234

==========================

==========================

Data:

67.5666

With Priority:

900

==========================

==========================

Data:

98.2753

With Priority:

90

==========================

==========================

Data:

67.111

With Priority:

78

==========================

==========================

Data:

78.3

With Priority:

67

==========================

==========================

Data:

78.456

With Priority:

9

==========================

==========================

Data:

729

With Priority:

78

==========================

==========================

Data:

4.555

With Priority:

7

==========================

Back of Queue

Testing HeapQ of strings

Quick test of empty print:

No items in the queue to print

Testing Enqueue Function:

Front of Queue

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

Back of Queue

Testing Copy Constructor: (should match above output)

Front of Queue

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

Back of Queue

Testing Overloaded Assignment: (should match above)

Front of Queue

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

Back of Queue

Preventing self assignment

Testing Enqueueing over and resizing array:

Front of Queue

==========================

Data:

jiminy

With Priority:

9000

==========================

==========================

Data:

Gerald

With Priority:

900

==========================

==========================

Data:

ariela

With Priority:

566

==========================

==========================

Data:

erika

With Priority:

101

==========================

==========================

Data:

ricky

With Priority:

91

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

skylar

With Priority:

30

==========================

==========================

Data:

Hailey

With Priority:

80

==========================

Back of Queue

Testing peek Function:

Working queue...

Front of Queue

==========================

Data:

jakob

With Priority:

900

==========================

==========================

Data:

armaya

With Priority:

90

==========================

==========================

Data:

harold

With Priority:

55

==========================

Back of Queue

Peeking off the top...

HeapObj at front of the queue:

==========================

Data:

jakob

With Priority:

900

==========================

Testing Dequeue Function:

Working queue

Front of Queue

==========================

Data:

jiminy

With Priority:

9000

==========================

==========================

Data:

Gerald

With Priority:

900

==========================

==========================

Data:

ariela

With Priority:

566

==========================

==========================

Data:

erika

With Priority:

101

==========================

==========================

Data:

ricky

With Priority:

91

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

skylar

With Priority:

30

==========================

==========================

Data:

Hailey

With Priority:

80

==========================

Back of Queue

Value that was returned...

jiminy

Resulting queue after clean up

Front of Queue

==========================

Data:

Gerald

With Priority:

900

==========================

==========================

Data:

erika

With Priority:

101

==========================

==========================

Data:

ariela

With Priority:

566

==========================

==========================

Data:

Ryan

With Priority:

100

==========================

==========================

Data:

ricky

With Priority:

91

==========================

==========================

Data:

jarvis

With Priority:

80

==========================

==========================

Data:

james

With Priority:

90

==========================

==========================

Data:

Hailey

With Priority:

80

==========================

==========================

Data:

skylar

With Priority:

30

==========================

Back of Queue

Quick note! All functions except print() will throw a string error if out of bounds of something is not correct. Everything else works in accordance with the rules of a priority queue.

**HeapQ.h:**

#ifndef \_HEAPQ\_H\_

#define \_HEAPQ\_H\_

#include <iostream>

#include <stdio.h>

template<class T>

struct HeapObj {

T data;

int priority;

HeapObj() {

data = {};

priority = 0;

}

HeapObj(T d, int p) {

data = d;

priority = p;

}

HeapObj<T>& operator=(const HeapObj<T>& rhs) {

if (this == &rhs) {

return \*this;

}

data = rhs.data;

priority = rhs.priority;

return \*this;

}

HeapObj(const HeapObj<T>& rhs) {

data = rhs.data;

priority = rhs.priority;

}

};

template<class T>

class HeapQ {

private:

HeapObj<T>\* arr;

int arrLength;

int heap\_size;

void increaseKey(int, int); // Increases the priority of the HeapObj

void increaseSize(); // Increases the size of the array to avoid overflow

void swap(HeapObj<T>&, HeapObj<T>&); // Swaps two given values in the priority queue

public:

HeapQ(); // Default Constructor

HeapQ(const HeapQ<T>&); // Copy Constructor

~HeapQ(); // Destructor

HeapQ<T>& operator=(const HeapQ<T>&); // Overloaded Assignment

T dequeue(); // Dequeue the first object in the queue and return it

void peek(); // Display the contents of the object in the front of the queue

void enqueue(T, int); // Enqueue an object into the queue with a specified priority

void print(); // Prints out the contents of the queue

void MaxHeapify(int); // Fixes violations in subtree rooted at A[i]

};

#endif

**HeapQ.cpp:**

#include "HeapQ.h"

/\*

\* increase-key Function:

\* Increases the priority of a specific HeapObj

\*/

template<class T>

void HeapQ<T>::increaseKey(int subscript, int newPriority) {

if (arr[subscript].priority > newPriority) {

throw "Priority error?";

}

arr[subscript].priority = newPriority;

while (subscript != 0 && arr[subscript].priority > arr[subscript / 2].priority) {

swap(arr[subscript], arr[subscript / 2]);

subscript /= 2;

}

}

/\*

\* increaseSize Function:

\* Pads the array size to avoid array overflow/seg faults

\*/

template<class T>

void HeapQ<T>::increaseSize() {

HeapObj<T>\* newArr = new HeapObj<T>[arrLength \* 2];

for (int i = 0; i < arrLength; i++) {

newArr[i] = arr[i];

}

delete [] arr;

arr = newArr;

arrLength \*= 2;

}

/\*

\* Default Constuctor:

\* Initializes the priority queue

\*/

template<class T>

HeapQ<T>::HeapQ() {

arr = new HeapObj<T>[4];

arrLength = 4;

heap\_size = -1;

}

/\*

\* Copy Constructor:

\* Copies a HeapQ over to another instance

\*/

template<class T>

HeapQ<T>::HeapQ(const HeapQ<T>& rhs) {

arr = new HeapObj<T>[rhs.arrLength];

if (rhs.heap\_size != -1) {

for (int i = 0; i <= rhs.heap\_size; i++) {

arr[i] = rhs.arr[i];

}

}

arrLength = rhs.arrLength;

heap\_size = rhs.heap\_size;

}

/\*

\* Destructor:

\* Deletes dynamically allocated data

\*/

template<class T>

HeapQ<T>::~HeapQ() {

delete [] arr;

}

/\*

\* Overloaded Assignment Operator:

\* Allows for assignment of a priority Queue into another priority Queue instance

\*/

template<class T>

HeapQ<T>& HeapQ<T>::operator=(const HeapQ<T>& rhs) {

if (this == &rhs) {

return \*this;

}

delete [] arr;

arr = new HeapObj<T>[rhs.arrLength];

if (rhs.heap\_size != -1) {

for (int i = 0; i <= rhs.heap\_size; i++) {

arr[i] = rhs.arr[i];

}

}

arrLength = rhs.arrLength;

heap\_size = rhs.heap\_size;

return \*this;

}

/\*

\* dequeue Function:

\* Dequeues the front object in the queue and then returns it

\*/

template<class T>

T HeapQ<T>::dequeue() {

if (heap\_size == -1) {

throw "No items in the queue";

}

HeapObj<T> rtnMe = arr[0];

arr[0] = arr[heap\_size];

heap\_size--;

MaxHeapify(0);

return rtnMe.data;

}

/\*

\* MaxHeapify Function:

\* Fixes violations in subtree rooted at A[i]

\*/

template<class T>

void HeapQ<T>::MaxHeapify(int i) {

int Lchild = 2 \* i + 1;

int Rchild = 2 \* i + 2;

int max = i;

if (Lchild <= heap\_size && arr[Lchild].priority > arr[i].priority) {

max = Lchild;

}

if (Rchild <= heap\_size && arr[Rchild].priority > arr[max].priority) {

max = Rchild;

}

if (max != i) {

swap(arr[i], arr[max]);

MaxHeapify(max);

}

}

/\*

\* swap HeapObj<T> Function:

\* Swaps two given values in the priority queue

\*/

template<class T>

void HeapQ<T>::swap(HeapObj<T>& first, HeapObj<T>& second) {

HeapObj<T> temp = first;

first = second;

second = temp;

}

/\*

\* peek Function:

\* Displays the contents of the object in the front of the queue

\*/

template<class T>

void HeapQ<T>::peek() {

if (heap\_size == -1) {

throw "There are no items in the queue";

}

std::cout << "HeapObj at front of the queue:" << std::endl;

std::cout << "==========================" << std::endl;

std::cout << "Data: " << std::endl;

std::cout << arr[0].data << std::endl;

std::cout << "With Priority: " << std::endl;

std::cout << arr[0].priority << std::endl;

std::cout << "==========================" << std::endl;

}

/\*

\* enqueue Function:

\* Enqueue's an object into the queue with a specified priority

\*/

template<class T>

void HeapQ<T>::enqueue(T obj, int priority) {

if (priority < 0) {

throw "Priority out of bounds";

}

HeapObj<T> insertMe(obj, 0);

if (heap\_size + 1 == arrLength) {

increaseSize();

}

arr[++heap\_size] = insertMe;

increaseKey(heap\_size, priority);

}

/\*

\* print Function:

\* Prints out the contents of the queue

\*/

template<class T>

void HeapQ<T>::print() {

if (heap\_size == -1) {

std::cout << "No items in the queue to print" << std::endl;

return;

}

std::cout << std::endl;

std::cout << "Front of Queue" << std::endl;

for (int i = 0; i <= heap\_size; i++) {

std::cout << "==========================" << std::endl;

std::cout << "Data: " << std::endl;

std::cout << arr[i].data << std::endl;

std::cout << "With Priority: " << std::endl;

std::cout << arr[i].priority << std::endl;

std::cout << "==========================" << std::endl;

}

std::cout << "Back of Queue" << std::endl;

std::cout << std::endl;

}