Raman Hliabovich - M12361188
Zachary Johnson - M12351633
Sean Sliter - M12282236

Lab 8 Report

In this lab two classes were created. One class was a Max Priority Queue and it was created using an ordered linked list. The class includes 3 functions which are Insert, Remove, and PrintQueue. The Insert function inserted integers into the queue and the queue is always in order after every insert with the max value being on the top of the queue. The advantage of having a max priority queue is that nodes with greater precedence move to the head of the queue, rather than always being added to the tail of a regular queue making the max values easier to remove. The disadvantage of having a max priority queue is that insertion time can take longer because nodes have to find their right place in the queue first.

The second class was called a Max Heap. The classes consisted of an Insert function, a Remove function, and a PrintHeap function. The Max Heap behaved similar to the Priority Queue with the max value always being on top but it used a tree structure instead of a linked list. The advantages of having a Max Heap over a regular binary tree is that the tree is always balanced and it is easier to remove and add to the heap. The disadvantage is that adding and removing from the heap can take longer than a regular binary tree causing poor performance.

Knowing how the Max Priority Queue and the Max Heap operates can help us excel in our future careers as software engineers. Know these data structures can help us design projects in a way that they operate at max efficiency.

<u>Task 1 Function Testing:</u> Right click on the Task1Main.cpp, go to properties and include it in the build. Exclude the rest of the mains.

Insert()

```
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value: 1

Value added to queue.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value: 3

Value added to queue.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value: 2

Value added to queue.
```

PrintQueue()

```
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 3
 3  2  1
```

Remove()

```
 3  2  1
Select an option:
1. Insert
2. Remove
3. Print
4. Exit
Enter: 2

3 has been removed.
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 3
 2  1
```

<u>Task 2 Function Testing:</u> Right click on the Task2Main.cpp, go to properties and include it in the build. Exclude the rest of the mains. The code was structured so that zeros in the heap represent empty nodes, so you can't use a zero for one of the values that you are inputting.

Insert()

```
Enter max size of heap (cannot be 0): 5

Size of heap: 5

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value(cannot be 0): 1

Value added to heap.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value(cannot be 0): 2

Value added to heap.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value(cannot be 0): 3

Value added to heap.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value(cannot be 0): 5

Value added to heap.

Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 1

Value(cannot be 0): 4

Value added to heap.
```

PrintHeap()

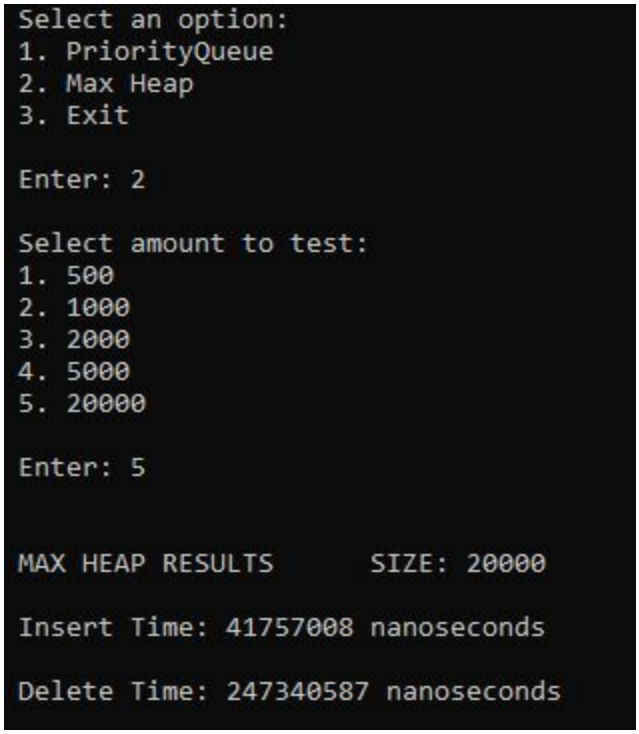
```
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 3

!!!!!! '0' represents an empty node in the heap !!!!!!

Max heap contents : 5  4  3  2  1
```

Remove()

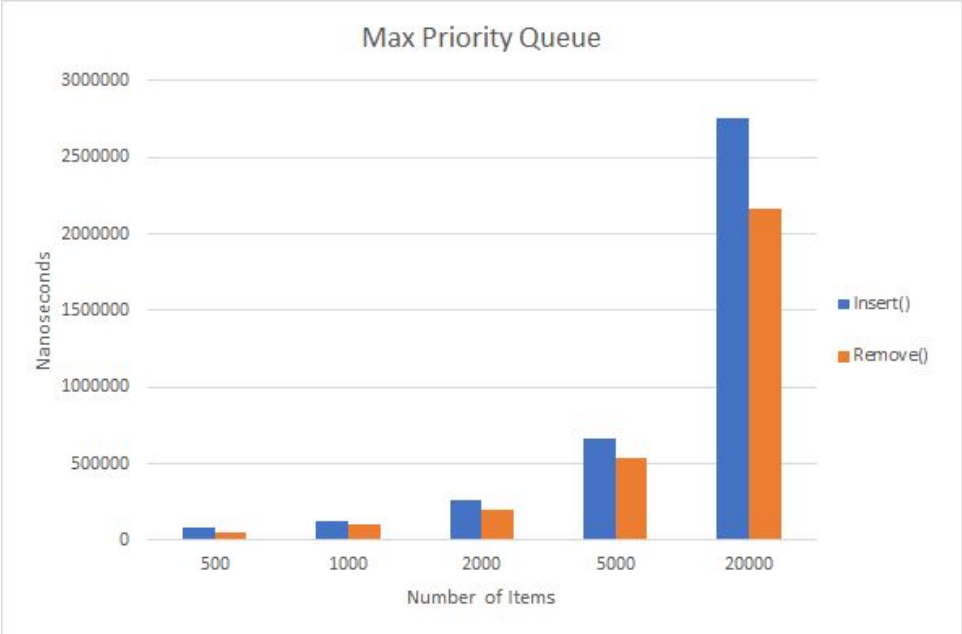
```
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 2

Enter value to remove: 3

3 has been removed.
Select an option:
1. Insert
2. Remove
3. Print
4. Exit

Enter: 3

!!!!!! '0' represents an empty node in the heap !!!!!!

Max heap contents : 5  4  2  1  0
```

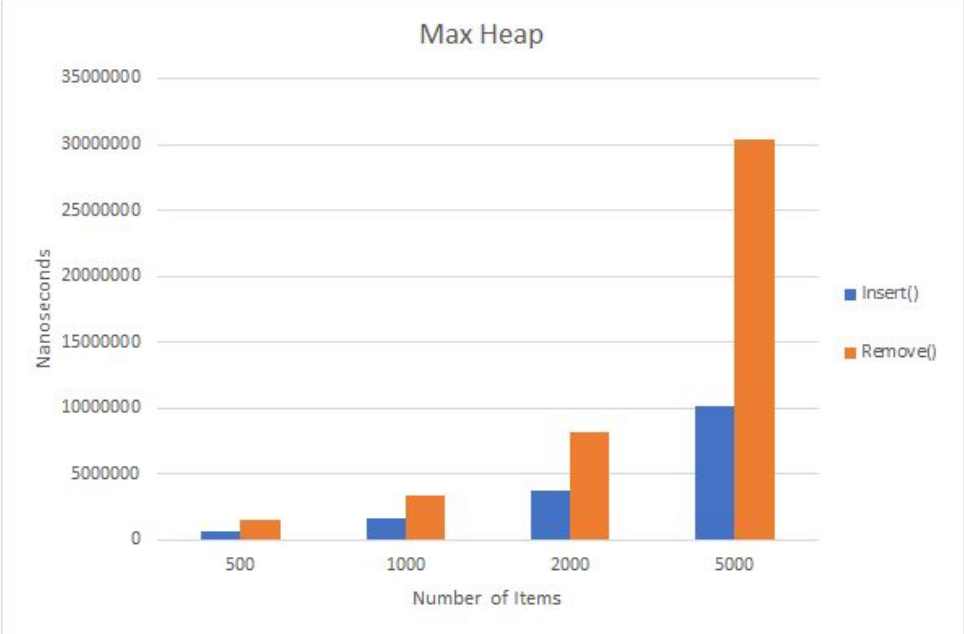
<u>Task 3 Time Testing:</u> Right click on the Task3Main.cpp, go to properties and include it in the build. Exclude the rest of the mains.

One of the test scenarios that was ran.



Graphs:





Discussion:

The Priority Queue performance was significantly better than the Max Heap performance. According to the Max Priority Queue graph inserting items took slightly more time than removing items, but the difference wasn't nearly as significant as the difference in times between inserting and removing items in the Max Heap. Also, inserting and removing times ranged from about 50k nanoseconds to about 2.7 million nanoseconds. According to the Max

Heap graph removing items almost took twice as long as inserting items and the times ranged from 1 million nanoseconds to about 40 million nanoseconds. Also, for the Max Heap graph I took out the 20000 items times because it was a huge outlier and made it hard to compare the time results for less items.

It was expected that the Max Priority Queue insertion and deletion times would be much faster than the Max Heap insertion and deletion times. It was also expected that the insertion and deletion times for the Max Priority Queue would be similar to each other, and it was expected the the deletion time for the Max Heap would be significantly more than the insertion time.