CSCE 221 - Programming Assignment 4 Report **(20 points)**

Due April 14, 2021

First Name: Jason Last Name: Hsu UIN: 428005390 User Name: jason3322 E-mailaddress: jason3322@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources					
People					
Web pages (provide URL)					
Printed material					
Other Sources					

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Jason Hsu Date 4/11/2021

1. The description of the assignment problem.

   The purpose of this assignment is to write a minimum priority queue with three kinds of data structure, vector, linked list and binary heaps. Also, learn to use object oriented programming by extending the parent class MPQ<T> with three child classes.

2. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   Vector: A vector is a one-dimensional array or, typically storing numbers. Vectors have fixed sizes, unlike lists and queues.

   List: A linked list is a linear data structure, in which the elements are not at contiguous memory locations. The elements in a linked list are linked using pointers.

   Binary heap: A binary heap is a heap data structure that takes the form of a binary tree. It’ s a common way of implementing priority queues.

   (b) Write about the ADTs implementation in C++ (for all the three MPQs).

   The vector is used to write the unsorted MPQ class. The linked list is used to write the sorted MPQ class. The binary heap class is used to write the binary heap MPQ class.

   (c) Describe algorithms used to solve the problem. For every MPQ (UnsortedMPQ, SortedMPQ and BinaryHeapMPQ), list the MPQ functions (remove_min(), is empty(), min(), and insert()) and provide their descriptions.

   - Unsort MPQ
     - Remove_min(): The function has a runtime of O(n), because the vector is unsorted, we have to visit every element in the vector to find the minimum and erase it. Same theory applies to Min() function.
     - Min(): The function has a runtime of O(n) as well.
     - Is_empty(): Check if the vector is empty, O(1)
     - Insert(): Because it’ s an unsorted vector, we can simply push back the inserted element to the end of the vector.
   - Sorted MPQ:
     - Remove_min(): The minimum is at the front of the list. Do pop front and return the minimum. Same theory applies to min(). O(1)
     - Min(): Same as Remove_min(), runtime of O(1)
     - Is_empty(): Check if the linked list is empty, O(1)
     - Insert(): Has a runtime of O(n). Due to the fact that it’ s a sorted list, we have to insert the element in the correct order.
   - Binary Heap MPQ:
     - insert(): we first push back the inserted data to the end of the vector. After that, we do a up heap to sort the binary heap to the correct order. O(logn)
     - Is_empty(): check if the size of the heap is 0, O(1)
     - Min(): O(1). Return the vector[0]
     - Remove_min(): Return the first element in the vector then do a down_heap to sort the vector from top to bottom. O(logn)

   (d) Show the time complexity analysis for the following. Time complexity analysis means providing a

basic runtime function/recurrence relation, **solution for recurrence relation with steps (wherever needed)** and a **Big-O** Notation:

i. **Best, worst, and average case** of each of the MPQ functions (remove min(), is empty(), min(), **and** insert()) for UnsortedMPQ. (Note: Some functions may have same runtimes for all the cases. In that case, write the answer only once and mention that the runtime applies to all the cases.).

- Unsort MPQ
  - Remove_min()
    - Best case: O(n)
    - Worst case: O(n)
    - Average case: O(n)
  - Min()
    - Best case: O(n)
    - Worst case: O(n)
    - Average case: O(n)
  - Is_empty():
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Insert():
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)

A. Provide an **example for best, worst, and average case** for UnsortedMPQ.

- Best case, worst case, and average case are the same for unsort MPQ , O(nlogn)
- [2,3,1,4,5,3,7]

ii. **Best, worst, and average case** of each of the MPQ functions (remove min(), is empty(), min(), **and** insert()) for SortedMPQ. (Note: Some functions may have same runtimes for all the cases. In that case, write the answer only once and mention that the runtime applies to all the cases).

- Sorted MPQ
  - Remove_min()
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Min()
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Is_empty():
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Insert():
    - Best case: O(1)
    - Worst case: O(n)
    - Average case: O(n)

A. Provide an **example for best, worst, and average case** for SortedMPQ.

- Best case, average case, and worst case are the same for sorted MPQ, O(nlogn)
- [1,2,5,7,10,9,8,3,2]

iii. **Best, worst, and average case** of each of the MPQ functions (remove min(), is empty(), min(), **and** insert()) for BinaryHeapMPQ. (Note: Some functions may have same runtimes for all the cases. In that case, write the answer only once and mention that the runtime applies to all the cases).

- Binary Heap MPQ
  - Remove_min()
    - Best case: O(logn)
    - Worst case: O(logn)
    - Average case: O(logn)
  - Min()
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Is_empty():
    - Best case: O(1)
    - Worst case: O(1)
    - Average case: O(1)
  - Insert():
    - Best case: O(logn)
    - Worst case: O(logn)
    - Average case: O(logn)

A. Provide an **example for best, worst, and average case** for BinaryHeapMPQ.

Best, worst, and average case are the same for BianryHeap MPQ, O(nlogn)
[10,988,23,43,65,12,90]

3. A C++ organization and implementation of the problem solution

   (a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

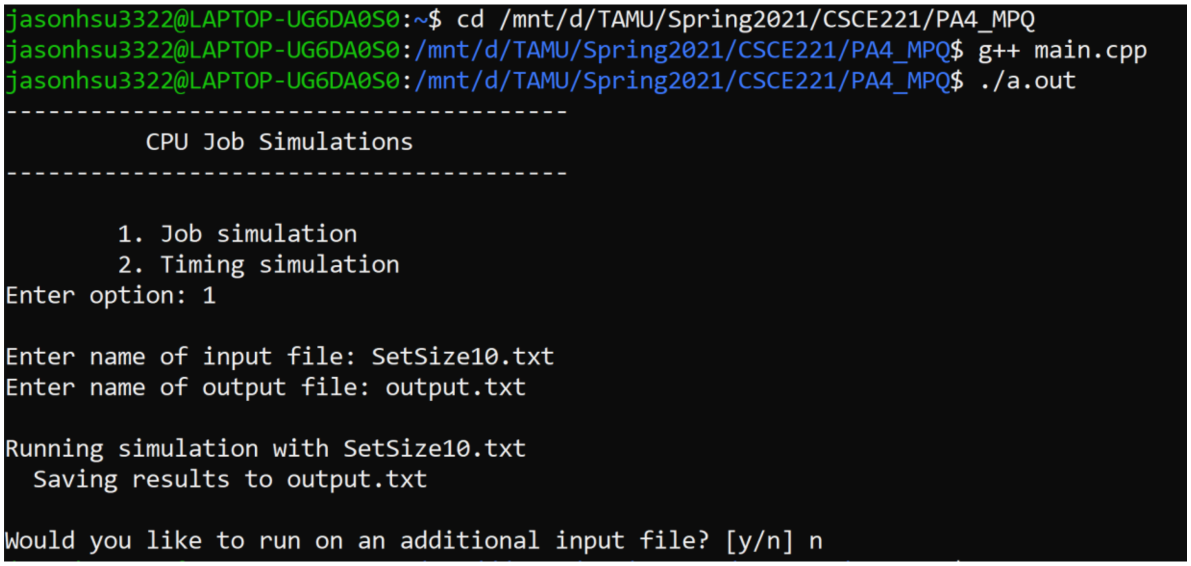
   One parent class MPQ is used to define the behavior of three MPQ subclasses. Four classes are used to by the program: Sort MPQ, Unsorted MPQ, Binary Heap, Binary Heap MPQ. A CPU job class is used to test the accuracy of all the previous classes.

   (b) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

   Templates is used in the program for the MPQ classes. Also, there is a parent class for the minimum priority queue. Three other MPQ classes inherit the parent class.

4. A user guide description how to navigate your program with the instructions how to:

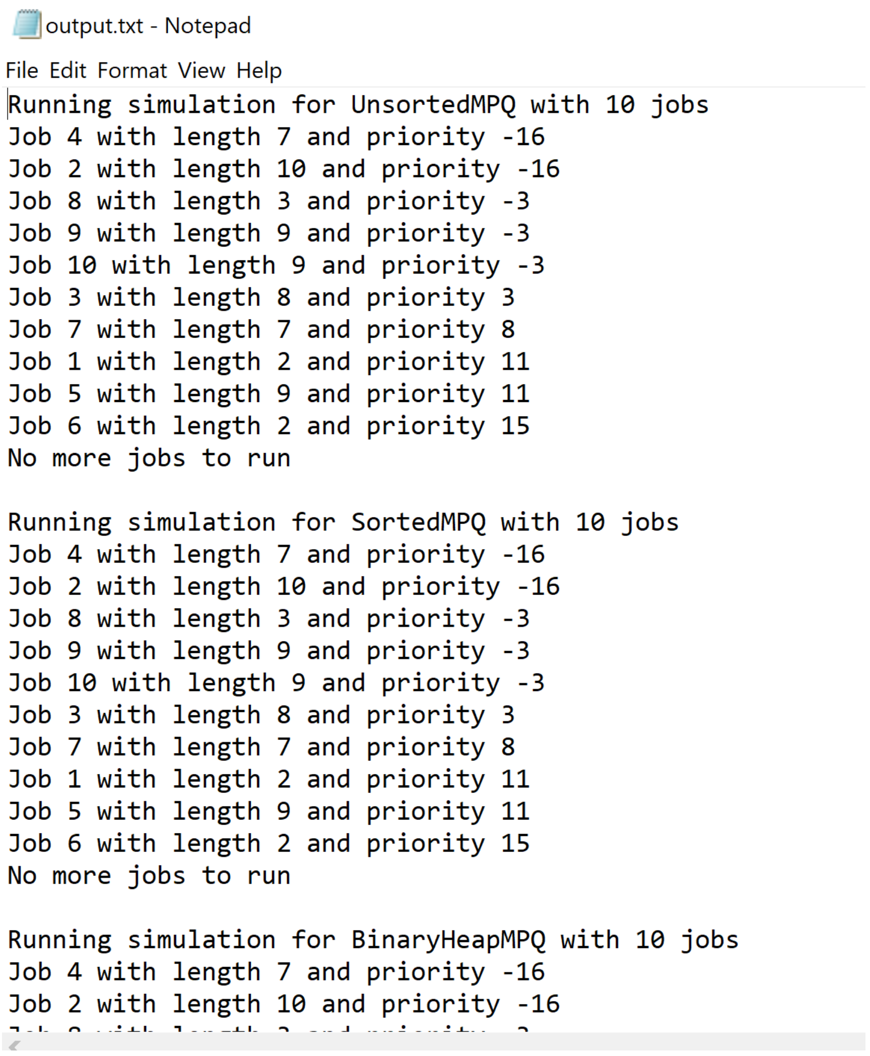
   (a) compile the program: specify the directory and file names, etc.



```
jasonhsu3322@LAPTOP-UG6DA0S0:~$ cd /mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ main.cpp
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
----------------------------------------
          CPU Job Simulations
----------------------------------------

        1. Job simulation
        2. Timing simulation
Enter option: 1

Enter name of input file: SetSize10.txt
Enter name of output file: output.txt

Running simulation with SetSize10.txt
  Saving results to output.txt

Would you like to run on an additional input file? [y/n] n
```

(b) run the program: specify the name of an executable file.

The output file as below



```
Running simulation for UnsortedMPQ with 10 jobs
Job 4 with length 7 and priority -16
Job 2 with length 10 and priority -16
Job 8 with length 3 and priority -3
Job 9 with length 9 and priority -3
Job 10 with length 9 and priority -3
Job 3 with length 8 and priority 3
Job 7 with length 7 and priority 8
Job 1 with length 2 and priority 11
Job 5 with length 9 and priority 11
Job 6 with length 2 and priority 15
No more jobs to run

Running simulation for SortedMPQ with 10 jobs
Job 4 with length 7 and priority -16
Job 2 with length 10 and priority -16
Job 8 with length 3 and priority -3
Job 9 with length 9 and priority -3
Job 10 with length 9 and priority -3
Job 3 with length 8 and priority 3
Job 7 with length 7 and priority 8
Job 1 with length 2 and priority 11
Job 5 with length 9 and priority 11
Job 6 with length 2 and priority 15
No more jobs to run

Running simulation for BinaryHeapMPQ with 10 jobs
Job 4 with length 7 and priority -16
Job 2 with length 10 and priority -16
```

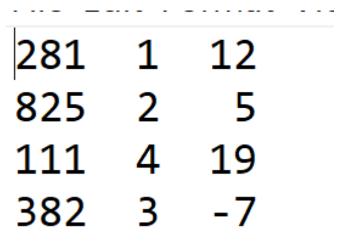
5. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

   Both the input files and the output files are txt files.

   (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

   An input file example is as follow. The first column would be the job ID. The second column is the job length, and the third column is the job priority. Each row represent a job.



```
281   1   12
825   2    5
111   4   19
382   3   -7
```

   (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

   The program could crash if the user feed in a file that does not follow the above format. It could also crash if the user does not feed in a txt file.

6. Provide types of exceptions and their purpose in your program (Answer only to the ones that are applicable for this assignment).

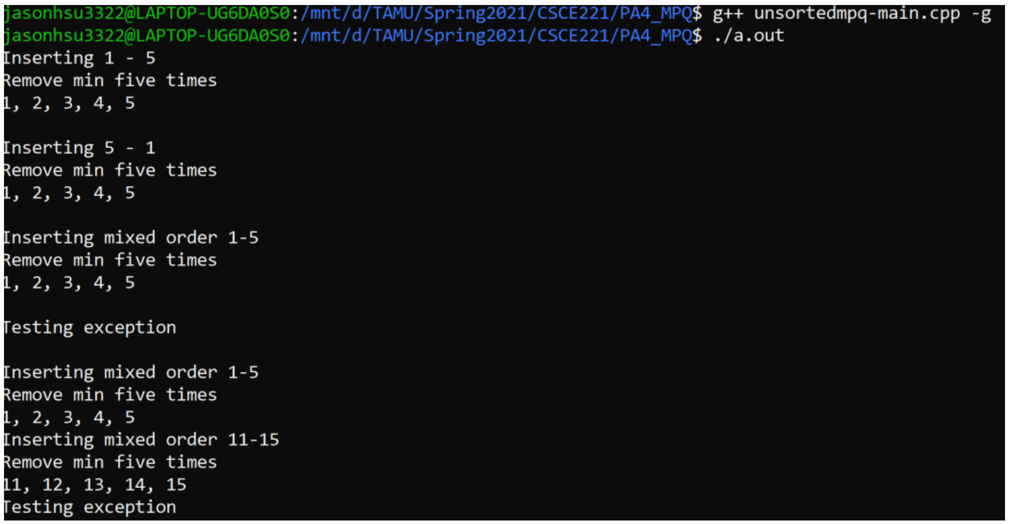
   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

   A possible logical exceptions would be the user feed in the incorrect file to the program.

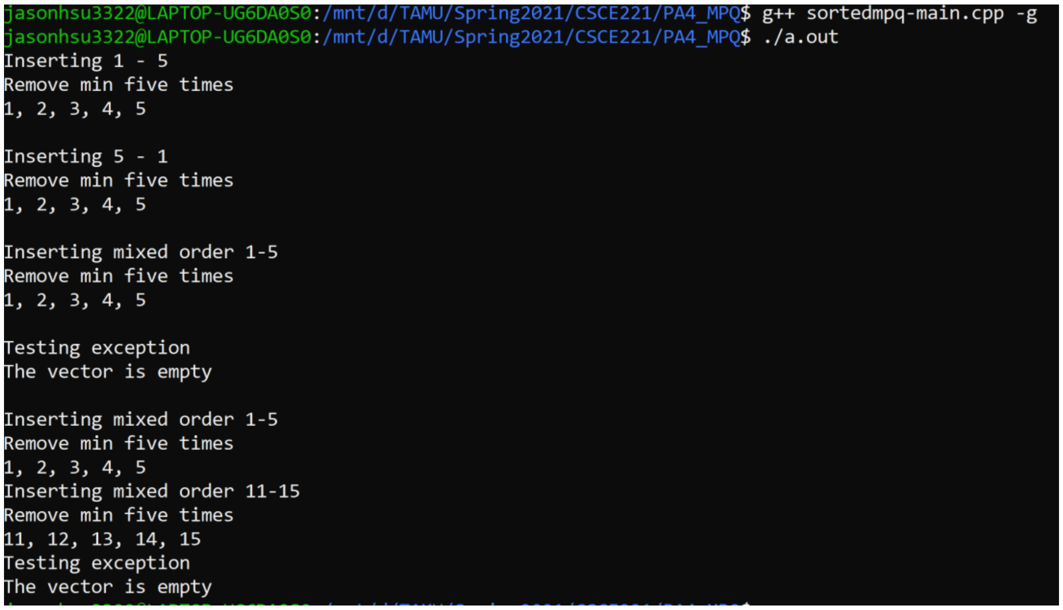
   (b) runtime exception (such as division by 0, etc.)

   A possible runtime exception would be the priority is not in the range of -20~19. The program required the priority of the job to be in this particular range.

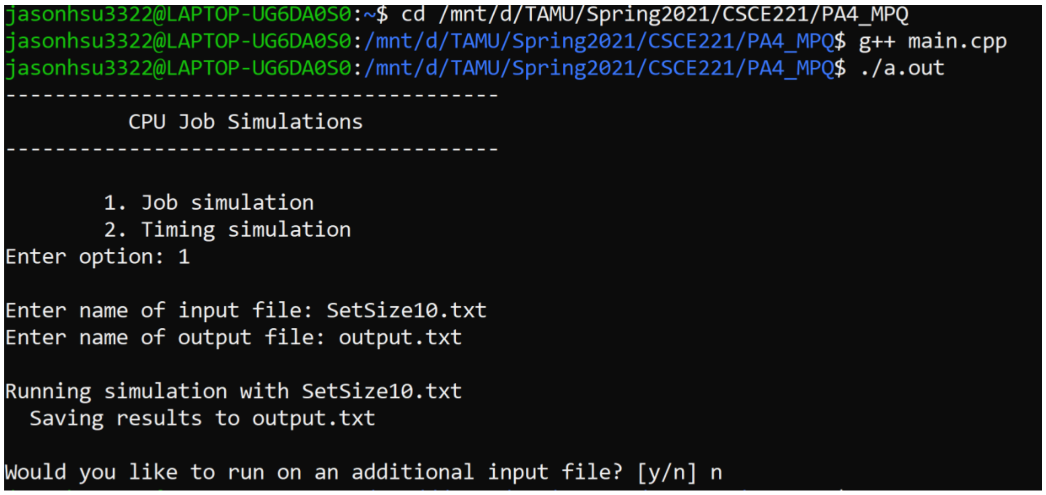
7. Include evidence of your testing by providing screenshots. Screenshots should show execution of the 5 main methods (unsortedmpq-main.cpp, sortedmpq-main.cpp, main.cpp, cpu-job-main.cpp, binaryheap-mpq-main.cpp).



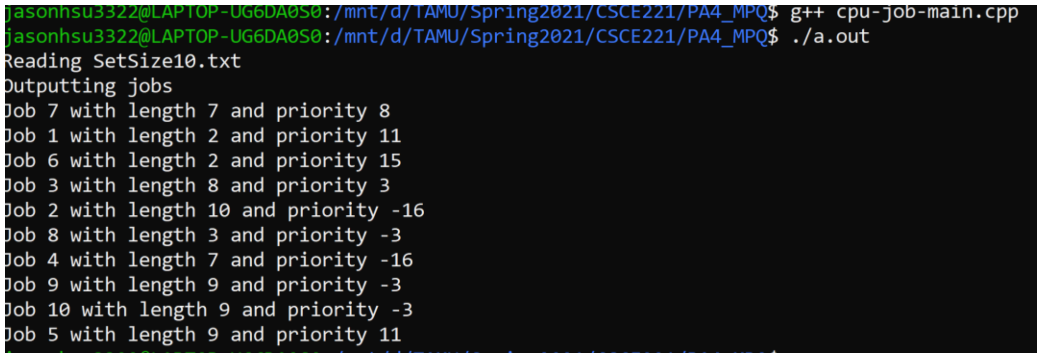
```
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ unsortedmpq-main.cpp -g
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
Inserting 1 - 5
Remove min five times
1, 2, 3, 4, 5

Inserting 5 - 1
Remove min five times
1, 2, 3, 4, 5

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5

Testing exception

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5
Inserting mixed order 11-15
Remove min five times
11, 12, 13, 14, 15
Testing exception
```



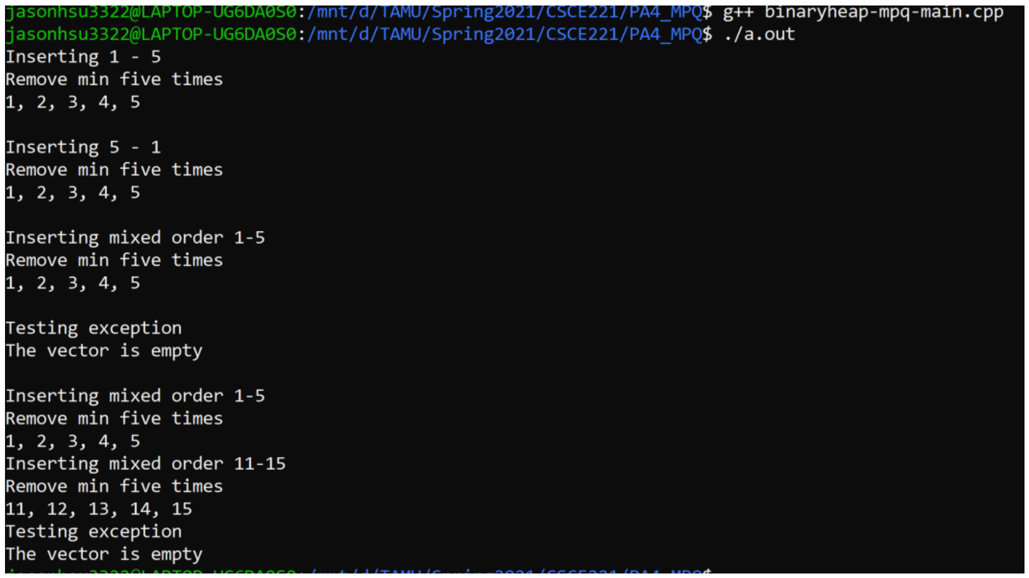
```
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ sortedmpq-main.cpp -g
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
Inserting 1 - 5
Remove min five times
1, 2, 3, 4, 5

Inserting 5 - 1
Remove min five times
1, 2, 3, 4, 5

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5

Testing exception
The vector is empty

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5
Inserting mixed order 11-15
Remove min five times
11, 12, 13, 14, 15
Testing exception
The vector is empty
```



```
jasonhsu3322@LAPTOP-UG6DA0S0:~$ cd /mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ main.cpp
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
----------------------------------------
          CPU Job Simulations
----------------------------------------

        1. Job simulation
        2. Timing simulation
Enter option: 1

Enter name of input file: SetSize10.txt
Enter name of output file: output.txt

Running simulation with SetSize10.txt
  Saving results to output.txt

Would you like to run on an additional input file? [y/n] n
```



```
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ cpu-job-main.cpp
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
Reading SetSize10.txt
Outputting jobs
Job 7 with length 7 and priority 8
Job 1 with length 2 and priority 11
Job 6 with length 2 and priority 15
Job 3 with length 8 and priority 3
Job 2 with length 10 and priority -16
Job 8 with length 3 and priority -3
Job 4 with length 7 and priority -16
Job 9 with length 9 and priority -3
Job 10 with length 9 and priority -3
Job 5 with length 9 and priority 11
```

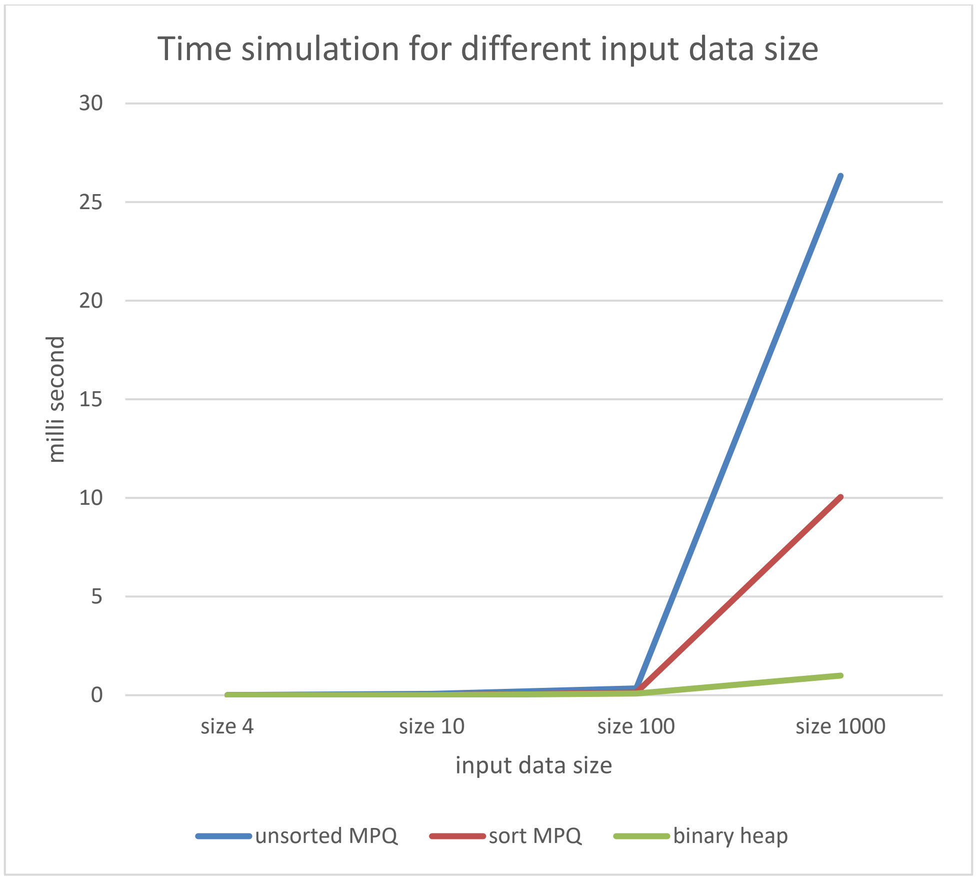


```
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ g++ binaryheap-mpq-main.cpp
jasonhsu3322@LAPTOP-UG6DA0S0:/mnt/d/TAMU/Spring2021/CSCE221/PA4_MPQ$ ./a.out
Inserting 1 - 5
Remove min five times
1, 2, 3, 4, 5

Inserting 5 - 1
Remove min five times
1, 2, 3, 4, 5

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5

Testing exception
The vector is empty

Inserting mixed order 1-5
Remove min five times
1, 2, 3, 4, 5
Inserting mixed order 11-15
Remove min five times
11, 12, 13, 14, 15
Testing exception
The vector is empty
```

8. Provide graphs and data tables of your CPU timing simulation results. Graph should be plotted for **runtime vs. input size**. The input sizes are 4, 10, 100, and 1,000. To obtain this data, compile and run main.cpp. Choose option "2. Timing Simulation". Provide the input file name (SetSize4.txt) and output filename. After execution, you will find the output file in "OutputFiles" folder. The timing for all the three MPQ implementations will be displayed. Fill it in the following table and plot it as a **graph**.

	Runtime		
Input Sizes	Unsorted MPQ	Sorted MPQ	Binary heap MPQ
4 (SetSize4.txt)	0.0082	0.0049	0.0057
10 (SetSize10.txt)	0.0757	0.0089	0.01
100 (SetSize100.txt)	0.3458	0.137	0.0869
1000 (SetSize1000.txt)	26.333	10.0487	0.9946

The unit of the time is in milli second.


Time simulation for different input data size
30
25
20
15
10
5
0
milli second
size 4
size 10
size 100
size 1000
input data size
unsorted MPQ
sort MPQ
binary heap