BLG335E, Analysis of Algorithms 1

Name:	İbrahim KARAHAN
Student Number:	150160550
CRN:	10983
Project 2	

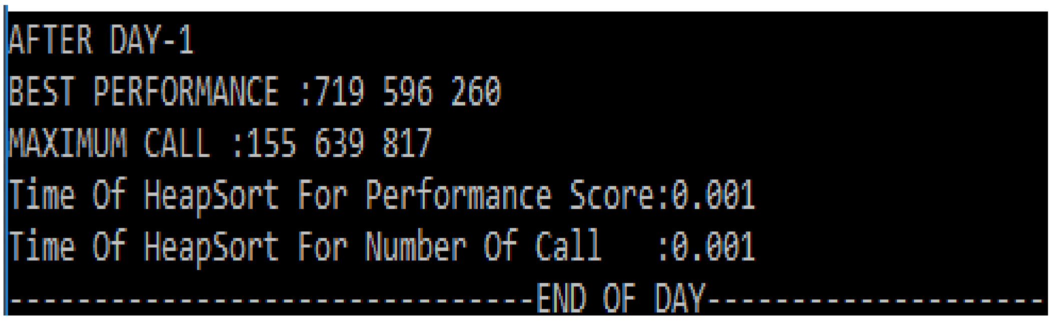
Contents

1. Part A

Project aim is writing a heap sort which sort the employee performance score and total number of call. Firstly, I read data from the file. In reading part I created an array for keeping the name of each file. I created this array for reading all file step by step. Then I created employee class for keeping value of each value. Employee class has features which are employee ıd, total number of calls, positive feedback calls, negative feedback calls and performance score. For calculating performance score I write a function. This function is used formula which given in the explanation of the homework. After calculation of each employee score I sent each employee value to employee array but controlling every employee object with increase key function and insert function. Firstly, each object sent to insert function in insert function call the increase key function. Increase key function controlled the object that exists already or not. If object exist, increase key function increased value of exist one and updated the object value. Increase key sent a Boolean value to insert function and insert function add new object to heap or not. After everything done in insert function heap updated with calling the maxheapifyPs function. In heap sort part, I used lecturer slide as a reference to my code and I used pseudo code of our lecture slide which is heap sort. I build four heap sort one of them sort the performance score according to the max/min value and the other one is sorting the number of call according to max/min value. I created extract max and extract min function this function separates the max and min value from the heap and decrease the heap size and height. After every extract operation I call insert function to add the extract element again heap. After every file is read from the file and sorted I print the best performance, worst performance, max number of call and min number of call.

2. Part B

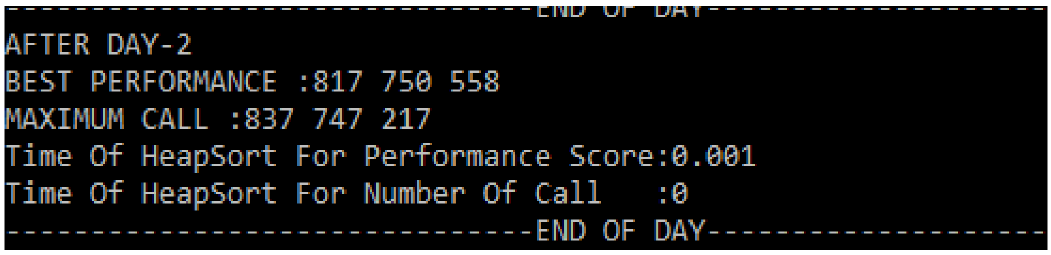
Firstly, I read day1 and built a heap according to vale of day1. In this heap day1 value is placing according to their performance score and number of call.



```
AFTER DAY-1
BEST PERFORMANCE :719 596 260
MAXIMUM CALL :155 639 817
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call   :0.001
-------------------------------END OF DAY--------------------
```

Figure-1

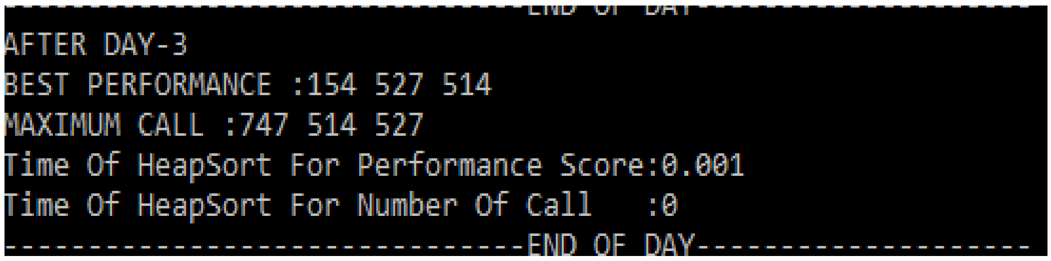
After reading day2 file, at this part heap updated according to day2 value.



```
AFTER DAY-2
BEST PERFORMANCE :817 750 558
MAXIMUM CALL :837 747 217
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
```

Figure-2

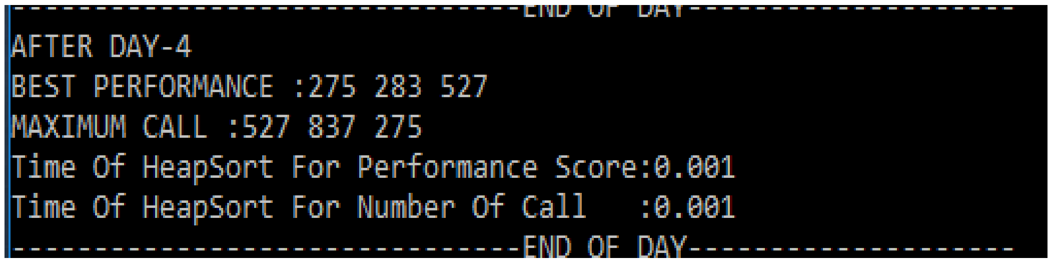
After reading day3 file, at this part heap updated according to day3 value and and sorted according to performance score and number of call.



```
AFTER DAY-3
BEST PERFORMANCE :154 527 514
MAXIMUM CALL :747 514 527
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
```

Figure-3

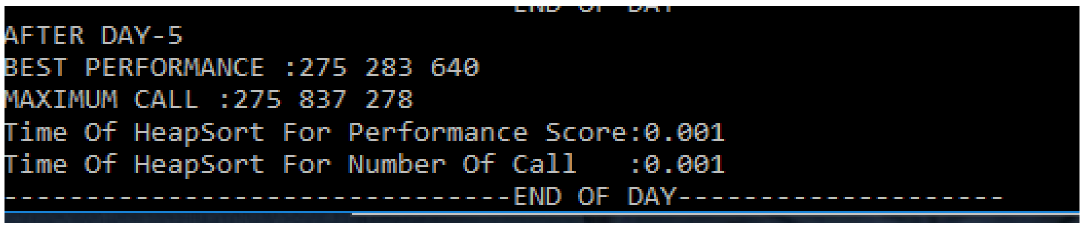
After reading day4 file, at this part heap updated according to day4 value and sorted according to performance score and number of call.



```
AFTER DAY-4
BEST PERFORMANCE :275 283 527
MAXIMUM CALL :527 837 275
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0.001
-------------------------------END OF DAY--------------------
```

Figure-4

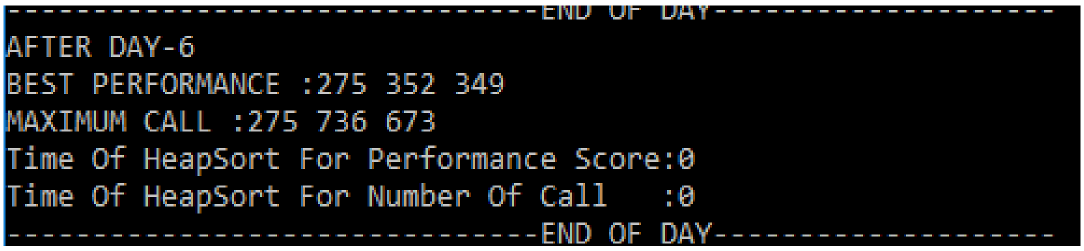
After reading day5 file, at this part heap updated according to day5 value and sorted according to performance score and number of call.



```
AFTER DAY-5
BEST PERFORMANCE :275 283 640
MAXIMUM CALL :275 837 278
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0.001
-------------------------------END OF DAY--------------------
```

Figure-5

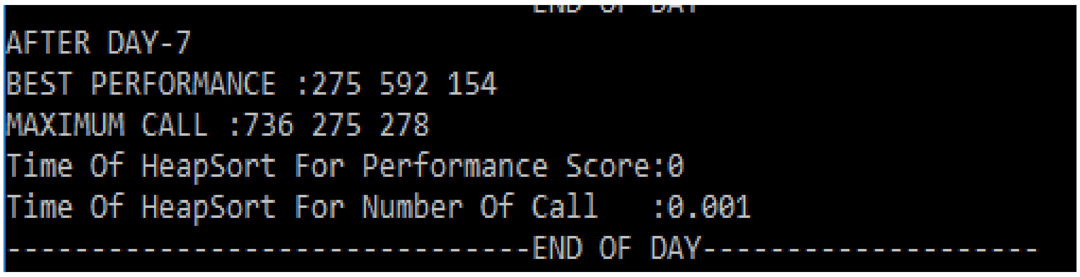
After reading day6 file, at this part heap updated according to day6 value and sorted according to performance score and number of call.



```
-------------------------------END OF DAY--------------------
AFTER DAY-6
BEST PERFORMANCE :275 352 349
MAXIMUM CALL :275 736 673
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
```

Figure-6

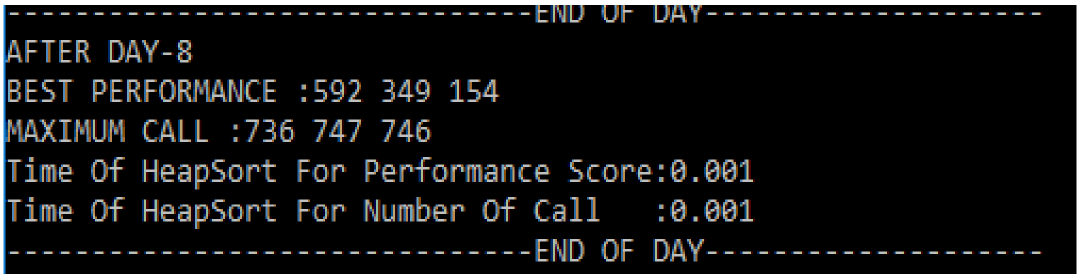
After reading day7 file, at this part heap updated according to day7 value and sorted according to performance score and number of call.



```
AFTER DAY-7
BEST PERFORMANCE :275 592 154
MAXIMUM CALL :736 275 278
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0.001
-------------------------------END OF DAY--------------------
```

Figure-7

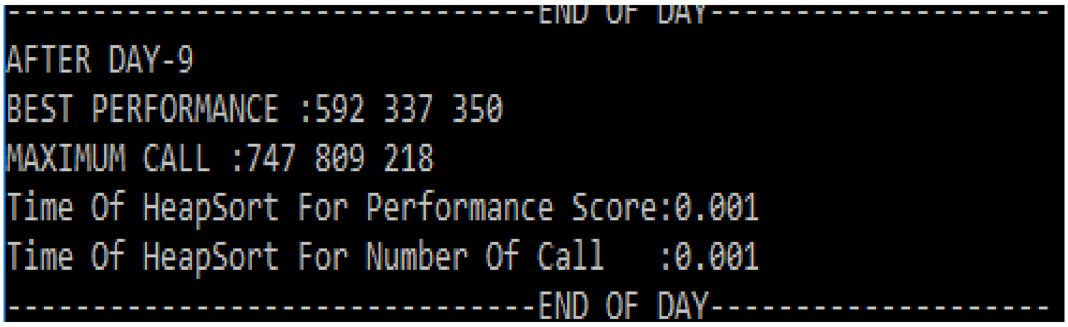
After reading day8 file, at this part heap updated according to day8 value and sorted according to performance score and number of call.



```
-------------------------------END OF DAY--------------------
AFTER DAY-8
BEST PERFORMANCE :592 349 154
MAXIMUM CALL :736 747 746
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0.001
-------------------------------END OF DAY--------------------
```

Figure-8

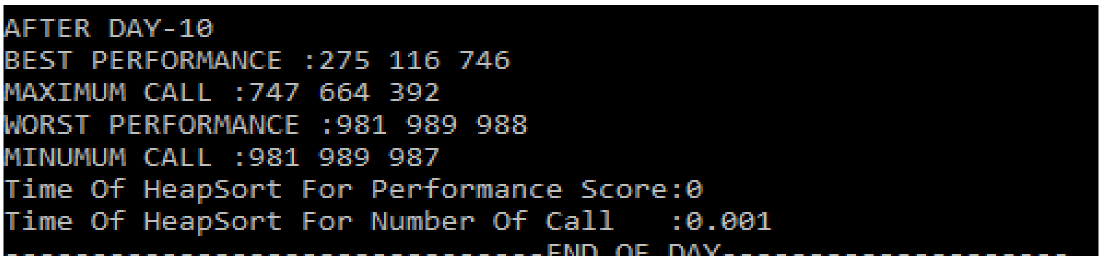
After reading day9 file, at this part heap updated according to day9 value and sorted according to performance score and number of call.



```
-------------------------------END OF DAY--------------------
AFTER DAY-9
BEST PERFORMANCE :592 337 350
MAXIMUM CALL :747 809 218
Time Of HeapSort For Performance Score:0.001
Time Of HeapSort For Number Of Call    :0.001
-------------------------------END OF DAY--------------------
```

Figure-9

After reading day10 file, at this part heap updated according to day10 value and sorted according to performance score and number of call. Also we can see the best employee after ten days.



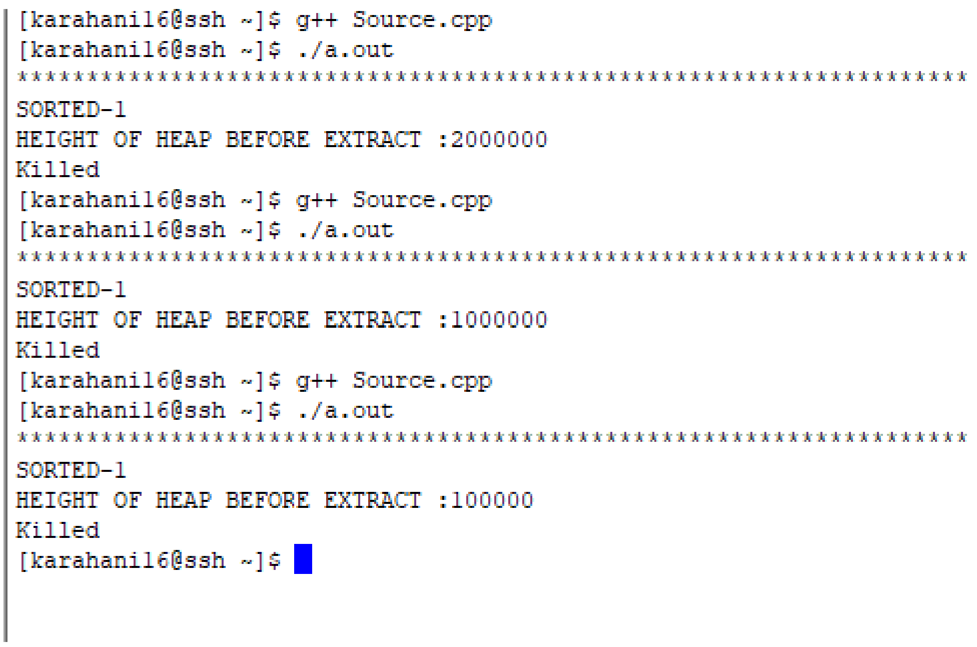
```
AFTER DAY-10
BEST PERFORMANCE :275 116 746
MAXIMUM CALL :747 664 392
WORST PERFORMANCE :981 989 988
MINUMUM CALL :981 989 987
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0.001
---------------------------------END OF DAY----------------------
```

Figure-10

3. Part C

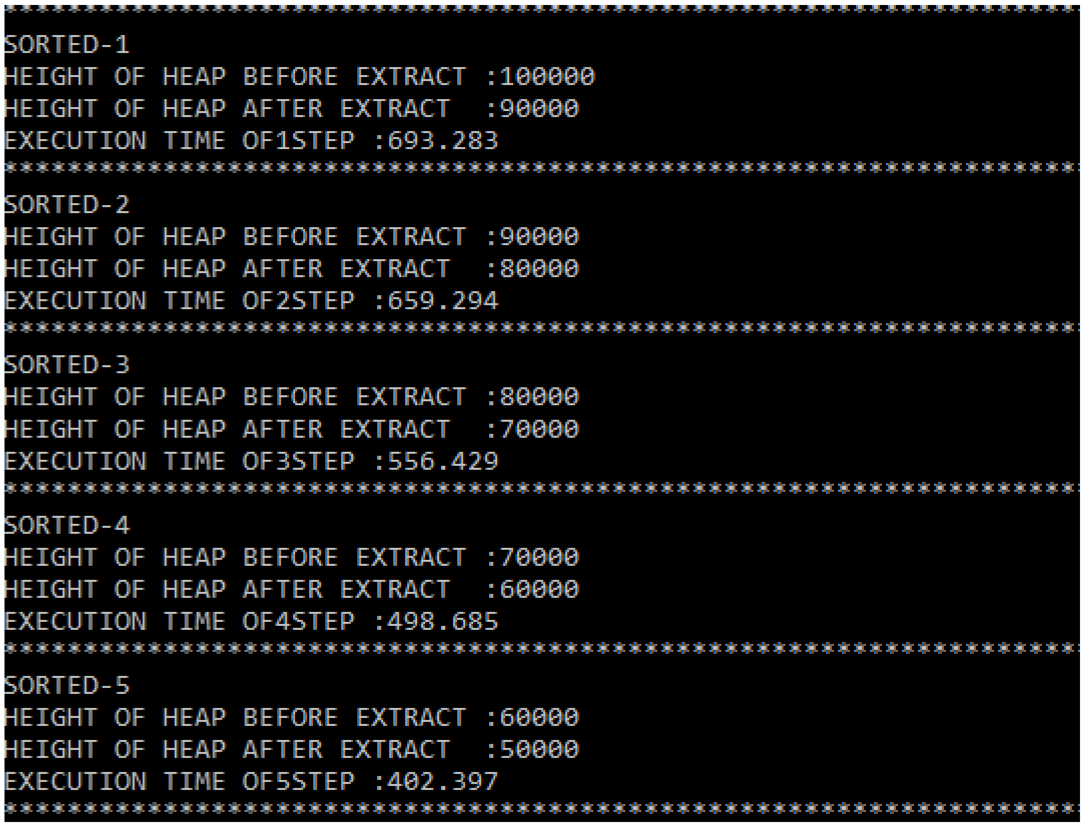
At this part, I used heap sort algorithm in the first one. Firstly, I read all data from numbers file then I sent all numbers to heap sort for sorting. Then I built a two loop outer loop is controlling the filename and printing the time and height of heap. Inner loop extracts the element from the heap step by step. Every extract operation I call heap sort for sorting the elements then extracts the other one.

I try to run my code with takes the 2M value but on the ITU server my process is killed because of the running time of my program so I decrease the number to 1M but I face with the same problem so at the end I determine 100K numbers so in here we can see the same result with 2M.



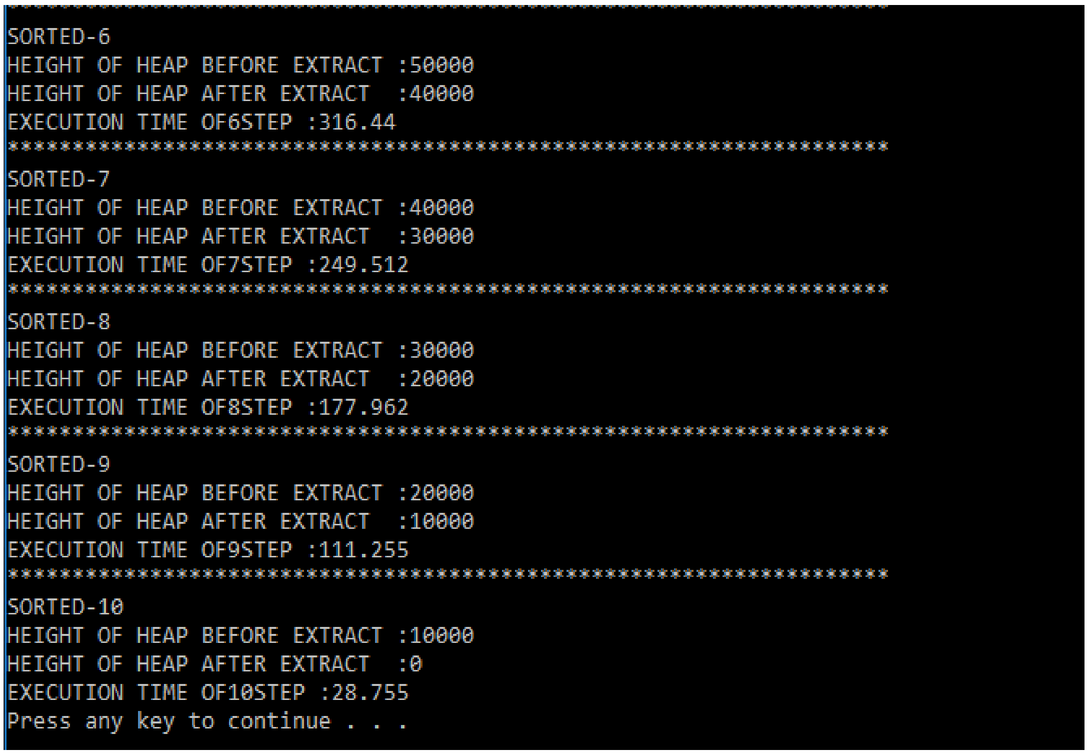
```
[karahanil6@ssh ~]$ g++ Source.cpp
[karahanil6@ssh ~]$ ./a.out
*********************************************************************
SORTED-1
HEIGHT OF HEAP BEFORE EXTRACT :2000000
Killed
[karahanil6@ssh ~]$ g++ Source.cpp
[karahanil6@ssh ~]$ ./a.out
*********************************************************************
SORTED-1
HEIGHT OF HEAP BEFORE EXTRACT :1000000
Killed
[karahanil6@ssh ~]$ g++ Source.cpp
[karahanil6@ssh ~]$ ./a.out
*********************************************************************
SORTED-1
HEIGHT OF HEAP BEFORE EXTRACT :100000
Killed
[karahanil6@ssh ~]$
```

Figure-11



```
*****************************************************************************
SORTED-1
HEIGHT OF HEAP BEFORE EXTRACT :100000
HEIGHT OF HEAP AFTER EXTRACT  :90000
EXECUTION TIME OF1STEP :693.283
*****************************************************************************
SORTED-2
HEIGHT OF HEAP BEFORE EXTRACT :90000
HEIGHT OF HEAP AFTER EXTRACT  :80000
EXECUTION TIME OF2STEP :659.294
*****************************************************************************
SORTED-3
HEIGHT OF HEAP BEFORE EXTRACT :80000
HEIGHT OF HEAP AFTER EXTRACT  :70000
EXECUTION TIME OF3STEP :556.429
*****************************************************************************
SORTED-4
HEIGHT OF HEAP BEFORE EXTRACT :70000
HEIGHT OF HEAP AFTER EXTRACT  :60000
EXECUTION TIME OF4STEP :498.685
*****************************************************************************
SORTED-5
HEIGHT OF HEAP BEFORE EXTRACT :60000
HEIGHT OF HEAP AFTER EXTRACT  :50000
EXECUTION TIME OF5STEP :402.397
*****************************************************************************
```

Figure-12



```
SORTED-6
HEIGHT OF HEAP BEFORE EXTRACT :50000
HEIGHT OF HEAP AFTER EXTRACT  :40000
EXECUTION TIME OF6STEP :316.44
*********************************************************************
SORTED-7
HEIGHT OF HEAP BEFORE EXTRACT :40000
HEIGHT OF HEAP AFTER EXTRACT  :30000
EXECUTION TIME OF7STEP :249.512
*********************************************************************
SORTED-8
HEIGHT OF HEAP BEFORE EXTRACT :30000
HEIGHT OF HEAP AFTER EXTRACT  :20000
EXECUTION TIME OF8STEP :177.962
*********************************************************************
SORTED-9
HEIGHT OF HEAP BEFORE EXTRACT :20000
HEIGHT OF HEAP AFTER EXTRACT  :10000
EXECUTION TIME OF9STEP :111.255
*********************************************************************
SORTED-10
HEIGHT OF HEAP BEFORE EXTRACT :10000
HEIGHT OF HEAP AFTER EXTRACT  :0
EXECUTION TIME OF10STEP :28.755
Press any key to continue . . .
```

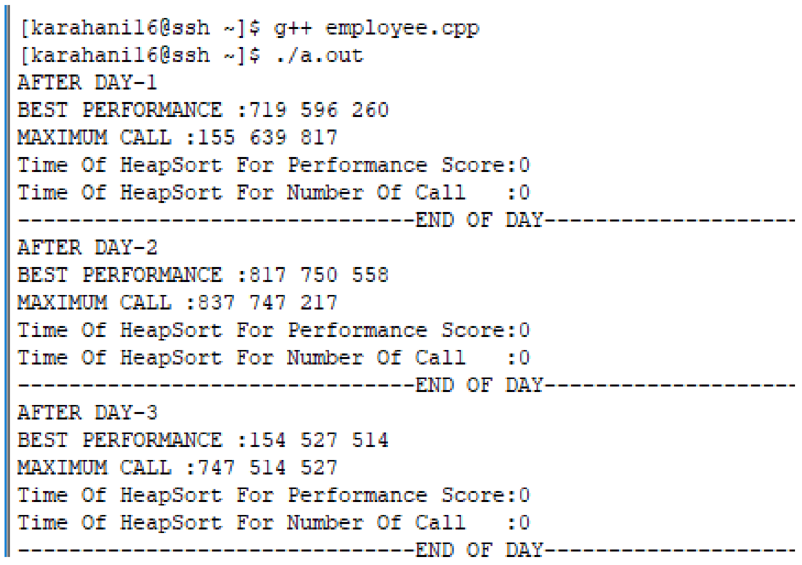
Figure-12

4. Conclusion

To sum up, I learn the implementation and compilation of heap sort. I observe the running time of heap sort with output of the program. Running time of the first part is increasing in every step because of the increasing heap size and the other part of running time is decreasing because of the decreasing at the heap size. I can say that running time of heap sort is in proportion with the log2 of heap size.

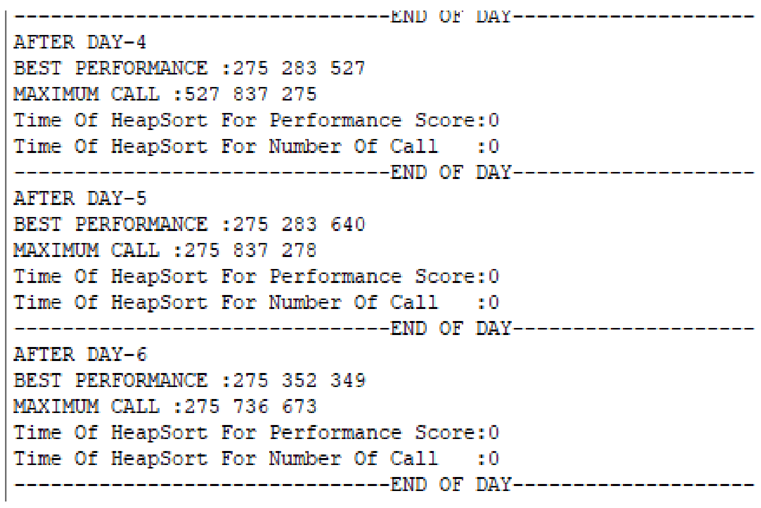
5. Compilation

I write two source files. One of them for sorting the employee and the other one is sorting the number in number.csv.



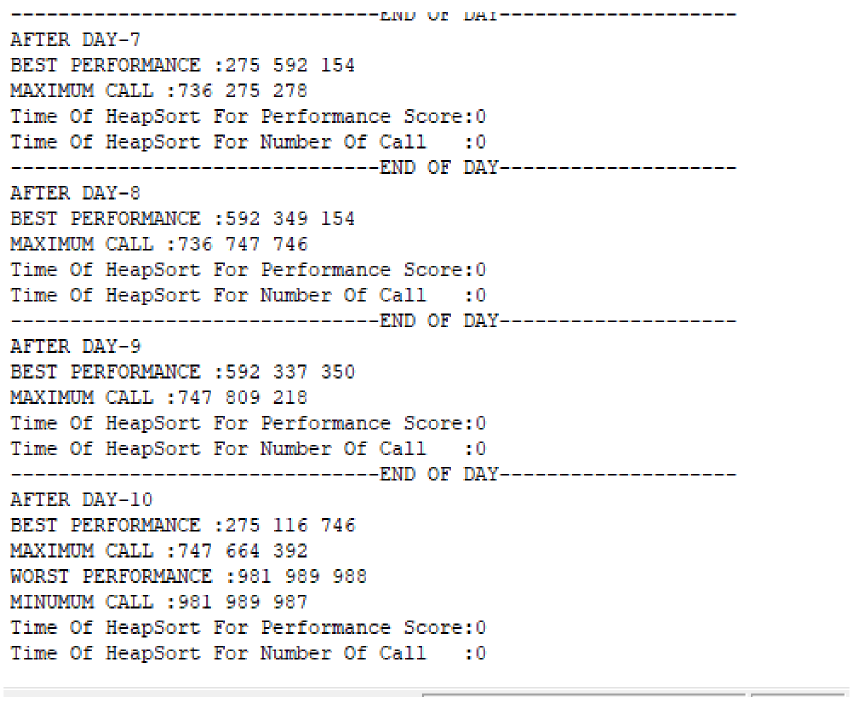
```
[karahanil6@ssh ~]$ g++ employee.cpp
[karahanil6@ssh ~]$ ./a.out
AFTER DAY-1
BEST PERFORMANCE :719 596 260
MAXIMUM CALL :155 639 817
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
AFTER DAY-2
BEST PERFORMANCE :817 750 558
MAXIMUM CALL :837 747 217
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
AFTER DAY-3
BEST PERFORMANCE :154 527 514
MAXIMUM CALL :747 514 527
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
```

Figure-13



```
-------------------------------END OF DAY--------------------
AFTER DAY-4
BEST PERFORMANCE :275 283 527
MAXIMUM CALL :527 837 275
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
AFTER DAY-5
BEST PERFORMANCE :275 283 640
MAXIMUM CALL :275 837 278
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
AFTER DAY-6
BEST PERFORMANCE :275 352 349
MAXIMUM CALL :275 736 673
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call    :0
-------------------------------END OF DAY--------------------
```

Figure-14



```
-------------------------------END OF DAY--------------------
AFTER DAY-7
BEST PERFORMANCE :275 592 154
MAXIMUM CALL :736 275 278
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call   :0
-------------------------------END OF DAY--------------------
AFTER DAY-8
BEST PERFORMANCE :592 349 154
MAXIMUM CALL :736 747 746
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call   :0
-------------------------------END OF DAY--------------------
AFTER DAY-9
BEST PERFORMANCE :592 337 350
MAXIMUM CALL :747 809 218
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call   :0
-------------------------------END OF DAY--------------------
AFTER DAY-10
BEST PERFORMANCE :275 116 746
MAXIMUM CALL :747 664 392
WORST PERFORMANCE :981 989 988
MINUMUM CALL :981 989 987
Time Of HeapSort For Performance Score:0
Time Of HeapSort For Number Of Call   :0
```

Figure-15