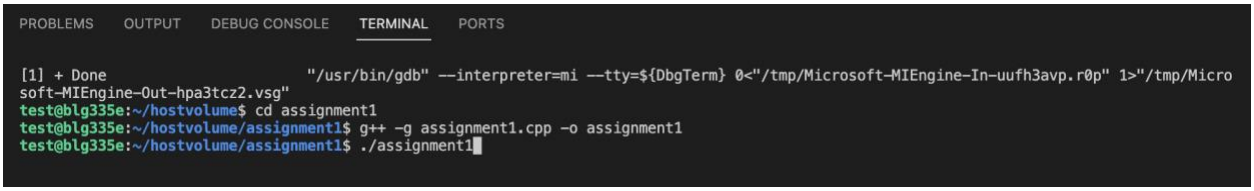
BLG335E – Assignment1 Report
Name: Ömer Malik Kalembaşı
Student ID: 150180112

a. How2Run

directory: cd assignment1
compile: g++ -g assignment1.cpp -o assignment1
run: ./assignment1




```
32  int partition(vector<float> &arr, vector<float> &id, int low, int high, int &numberOfPartitions){
33
34      numberOfPartitions++;
35
36      if(high > low ){
37          int i = low;
38          int j = low;
39          float pivot = arr[high];
40          float temp = 0;
41
42          while(arr[i] <= pivot && arr[j] <= pivot ){
43
44              if(i < high && j <high){
45                  j++;
46                  i++;
47              }
48              else{
49                  break;
50              }
51          }
52
53          while(arr[j] > pivot ){
54              if(j < high){
55                  j++;
56              }
57              else{
58                  break;
59              }
60
61              temp = arr[i];  //swap avg rating
62              arr[i] = arr[j];
63              arr[j] = temp;
64
65              temp = id[i];   //swap id
66              id[i] = id[j];
67              id[j] = temp;
68
69              if(arr[i] <= pivot){
70                  if(i < high){
71                      i++;
72                  }
73                  else{
74                      break;
75                  }
```

```
            }
        }
    }

    temp = 0;

    if(j == high){
        temp = arr[i];  //swap avg rating
        arr[i] = arr[high];
        arr[high] = temp;

        temp = id[i];   //swap id
        id[i] = id[j];
        id[j] = temp;
    }
    return i;
  }
}

void quickSort(vector<float> &arr, vector<float> &id, int low, int high, int &numberOfPartitions, int &numberOfSwaps){

    numberOfSwaps++;

    if(high > low ){

        int p = partition(arr, id, low, high, numberOfPartitions);
        quickSort(arr, id, low, p-1, numberOfPartitions, numberOfSwaps);
        quickSort(arr, id, p, high, numberOfPartitions, numberOfSwaps);
    }
}
```

b. Sorted results in .txt

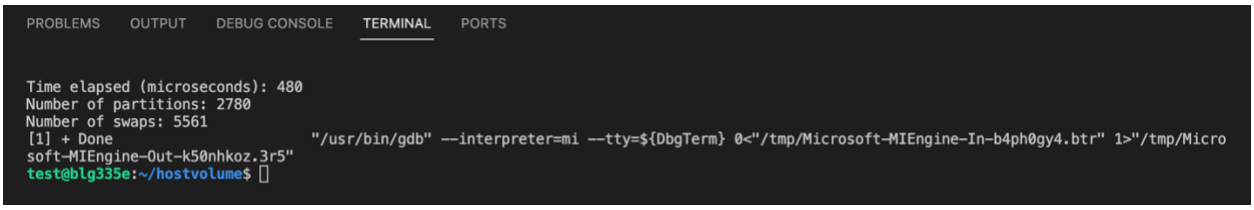
assignment1 > sorted_books.txt

```
799,0
1302,0
1537,0
1549,0
2442,0
3351,0
3479,0
3852,0
5720,0
5934,0
6625,0
7848,0
9337,0
9338,0
10200,0
10536,0
12712,0
15186,0
16806,0
17054,0
19257,0
19668,0
19858,0
21534,0
21536,0
21538,0
```

c. Time elapsed (microseconds), number of partitions, number of swaps

Time elapsed (microseconds), number of partitions, number of swaps (half

Time elapsed (microseconds), number of partitions, number of swaps (quarter)



d.

Worst case:

This happens when the array is already sorted in reverse order and we choose the pivot as the largest or smallest element. With each partition move, we split the array in half; one element and the rest, until two elements remain.

$n + (n-1) + (n-2) + (n-3) + (n-4) + ... + 2 = n*(n+1)/2 + 1 = o(n^2)$

$T(n) = o(n^2)$

Best case:

Best case occurs when we select mean as pivot. In this case recursion occurs like logn. Array will be divided in brances of equal size. Thus, height of the tree will be minimum. In each branch, we will traverse to the all elements, where n comes.

T(n) = n*logn

Average case:

Average case calculated by taking average of all comlexities, when occures all elements divided properly, not nearly descending or ascending.

T(n) = n*logn

e. The worst case happens when the elements of the array are almost sorted. To avoid the worst case, we choose the element farthest from the largest or smallest value as the pivot. It can be mean, median, or just random selection.