BLG 336E - Analysis of Algorithms II
2019/2020 Spring
Final Project
Question 1

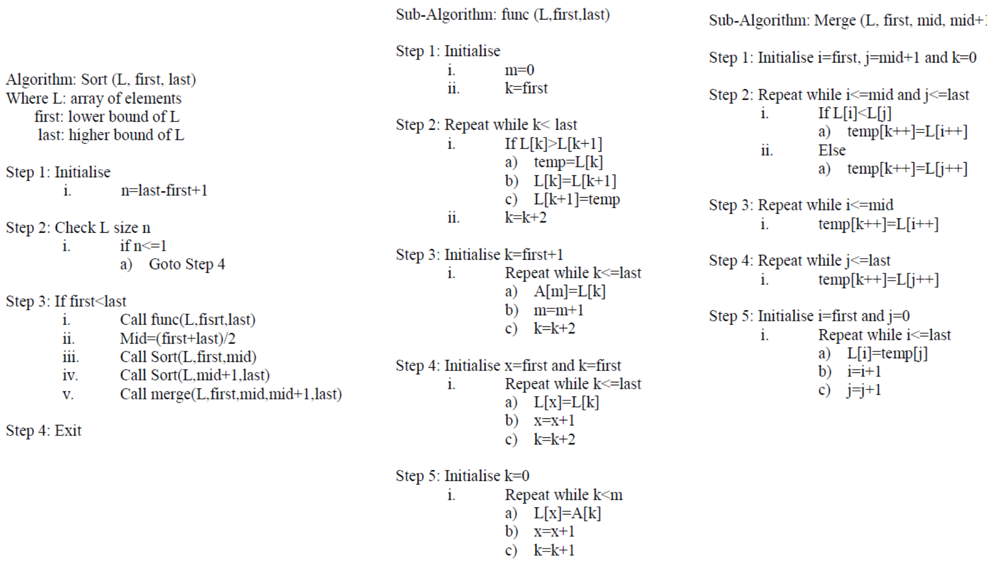
- You should write all your code in C++ language. Your code should be run with the command line arguments specified for each question.
- Your code should be able to be compiled with default g++ compiler and run under Ubuntu OS. **Even if you are writing your code on a different OS, you should check it via ITU SSH.**
- This is a Final Course Project Assignment, cheating is absolutely unethical and morally unacceptable. It will be punished by a negative grade. Also disciplinary actions will be taken.
- For every part of the homework, programs should be run with different command line arguments. **The codes not using these arguments or giving output in a different layout will not be graded**.

1 - Divide & Conquer (25 pts)

[**10 pts**] In [1], the authors asserted that execution time of Merge Sort can be decreased with a modification on merging operation. Thus, they created the Enhanced Merge Sort algorithm. The main difference is dividing the array into two sub-lists according to each element's odd-even positions before the recursive sorting operation. An example operation is given in Figure 1. Pseudocode of the algorithm is as given in Figure 1 where the main contribution is the **func** function.

Using the skeleton code **q1_1.cpp** implement the EMS algorithm. Your code should take the filename as the first argument. An example output for **q1_test1.txt** is as given below. Here first the unsorted list, then the divisions and lastly the sorted list is displayed.

```
g++ q1_1.cpp -o q1_1
./q1_1 test1.txt
6, 1, 10, 4, 8, 5, 7, 9
1, 4, 5, 7
```



```
Algorithm: Sort (L, first, last)
Where L: array of elements
      first: lower bound of L
      last: higher bound of L

Step 1: Initialise
        i.      n=last-first+1

Step 2: Check L size n
        i.      if n<=1
                a)  Goto Step 4

Step 3: If first<last
        i.      Call func(L,fisrt,last)
        ii.     Mid=(first+last)/2
        iii.    Call Sort(L,first,mid)
        iv.     Call Sort(L,mid+1,last)
        v.      Call merge(L,first,mid,mid+1,last)

Step 4: Exit
```

```
Sub-Algorithm: func (L,first,last)

Step 1: Initialise
        i.      m=0
        ii.     k=first

Step 2: Repeat while k< last
        i.      If L[k]>L[k+1]
                a)  temp=L[k]
                b)  L[k]=L[k+1]
                c)  L[k+1]=temp
        ii.     k=k+2

Step 3: Initialise k=first+1
        i.      Repeat while k<=last
                a)  A[m]=L[k]
                b)  m=m+1
                c)  k=k+2

Step 4: Initialise x=first and k=first
        i.      Repeat while k<=last
                a)  L[x]=L[k]
                b)  x=x+1
                c)  k=k+2

Step 5: Initialise k=0
        i.      Repeat while k<m
                a)  L[x]=A[k]
                b)  x=x+1
                c)  k=k+1
```

```
Sub-Algorithm: Merge (L, first, mid, mid+1

Step 1: Initialise i=first, j=mid+1 and k=0

Step 2: Repeat while i<=mid and j<=last
        i.      If L[i]<L[j]
                a)  temp[k++]=L[i++]
        ii.     Else
                a)  temp[k++]=L[j++]

Step 3: Repeat while i<=mid
        i.      temp[k++]=L[i++]

Step 4: Repeat while j<=last
        i.      temp[k++]=L[j++]

Step 5: Initialise i=first and j=0
        i.      Repeat while i<=last
                a)  L[i]=temp[j]
                b)  i=i+1
                c)  j=j+1
```

Figure 1: Enhanced Merge Sort algorithm.

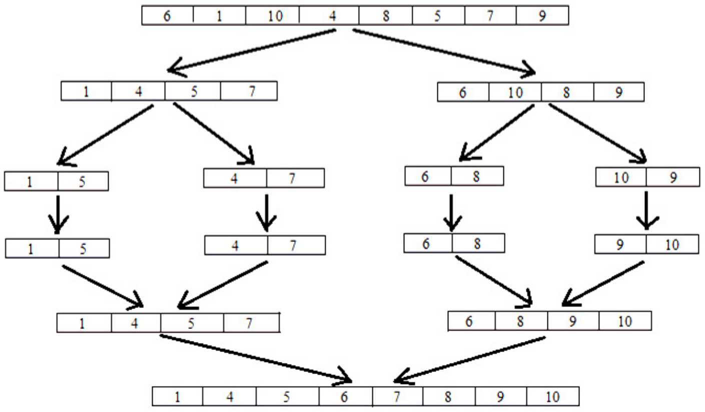


Figure 2: Enhanced Merge Sort on a list

```
1, 5
4, 7
6, 10, 8, 9
6, 8
10, 9
1, 4, 5, 6, 7, 8, 9, 10
```

[10 pts] Min-max sorting is a sorting algorithm which also works in a divide & conquer manner. Basically it can be summarised as follows;

- Find minimum value of the array **in a D&C way**.
- Find maximum value of the array **in a D&C way**.
- Swap these items with the first and last items.
- Iteratively follow the same procedure on the remaining part of the array.

Change the code you used in the first part so that for left sub-arrays it uses Min-Max Sorting and right sub-arrays it uses EMS. Output for this part for the same input file is given below.

```
g++ q1_2.cpp -o q1_2
./q1_2 test1.txt
6, 1, 10, 4, 8, 5, 7, 9
Min_max unsorted: 1, 4, 5, 7
Min: 1 Max: 7
Min: 4 Max: 5
Min_max sorted: 1, 4, 5, 7
6, 10, 8, 9
Min_max unsorted: 6, 8
Min: 6 Max: 8
Min_max sorted: 6, 8
10, 9
Min_max unsorted: 9
Min_max sorted: 9
1, 4, 5, 6, 7, 8, 9, 10
```

[5 pts] Make the complexity analysis of this newly frankenstained algorithm.

Bibliography

[1] S. Paira, S. Chandra, and S. S. Alam, "Enhanced merge sort-a new approach to the merging process," *Procedia Computer Science*, vol. 93, pp. 982–987, 2016. 1