

# Data Structures and Algorithms

## Lab Report

### Lab08



Group Members Name & Reg #:	<b><u>Muhammad Haris Irfan</u></b> <b>(FA18-BCE-090)</b>
Class	Data Structures and Algorithms CSC211 ( <b>BCE-3B</b> )
Instructor's Name	Dilshad Sabir

# In Lab Tasks

## Task:1

Complete the functions for Selection Sort and Insertion Sort

## Solution:

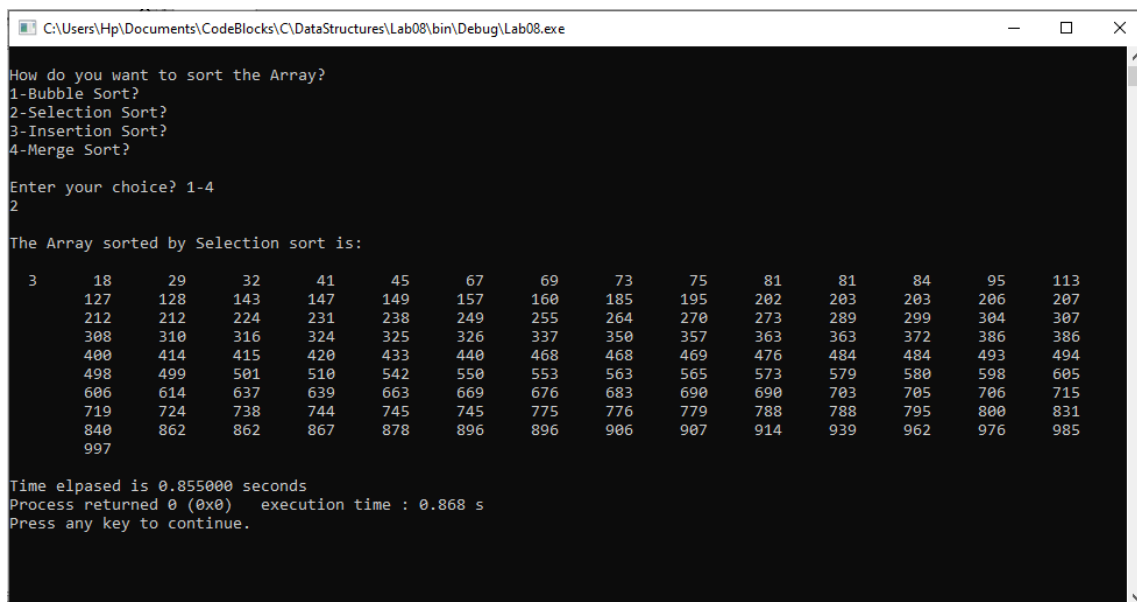
The code is shown below,

### Selection Sort:

```
int min_idx;
int temp;
for (int i = 0; i < arr_size-1; i++)
{
    min_idx = i;
    for (int j = i+1; j < arr_size; j++)
        if (*(ptr_array+j) < *(ptr_array+min_idx))
            min_idx = j;

    temp=*(ptr_array+i);
    *(ptr_array+i)=*(ptr_array+min_idx);
    *(ptr_array+min_idx)=temp;
}
```

The Result of the following code is attached below:



```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab08\bin\Debug\Lab08.exe
How do you want to sort the Array?
1-Bubble Sort?
2-Selection Sort?
3-Insertion Sort?
4-Merge Sort?
Enter your choice? 1-4
2
The Array sorted by Selection sort is:
3   18   29   32   41   45   67   69   73   75   81   81   84   95   113
   127  128  143  147  149  157  160  185  195  202  203  203  206  207
   212  212  224  231  238  249  255  264  270  273  289  299  304  307
   308  310  316  324  325  326  337  350  357  363  363  372  386  386
   400  414  415  420  433  440  468  468  469  476  484  484  493  494
   498  499  501  510  542  550  553  563  565  573  579  580  598  605
   606  614  637  639  663  669  676  683  690  690  703  705  706  715
   719  724  738  744  745  745  775  776  779  788  788  795  800  831
   840  862  862  867  878  896  896  906  907  914  939  962  976  985
   997

Time elapsed is 0.855000 seconds
Process returned 0 (0x0)   execution time : 0.868 s
Press any key to continue.
```

## Insertion Sort:

```
int i, key, j;
for (i = 1; i < arr_size; i++) {
    key = *(ptr_array+i);
    j = i - 1;

    while (j >= 0 && *(ptr_array+j) > key) {
        *(ptr_array+j+1) = *(ptr_array+j);
        j = j - 1;
    }
    *(ptr_array+j+1) = key;
}
```

## The Result of the following code is attached below:

-----

## **Task:2**

**Empirically compute the execution times for the three sorting methods**

## **Solution:**

We calculated the execution time of each type, by varying the Data size, the calculations are filled in the table below.

Data Size	Bubble Sort	Selection Sort	Insertion Sort	Merge Sort
16	1.14s	1.74s	0.90s	0.74s
128	0.90s	0.84s	0.58s	0.47s
1024	1.26s	1.46s	1.10s	0.64s
16384	2.86s	2.55s	2.43s	1.78s
131072	55.18s	29.88s	21.12s	11.7s

-----

# Post Lab Task.

## Task 3:

Complete the Merge Sort Function and empirically determine its time complexity.

## Solution

The code is shown below,

```
void merge(int * ptr_array, int l, int m, int r)
{
    /*** Complete this function ***/
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = *(ptr_array+l + i);
    for (j = 0; j < n2; j++)
        R[j] = *(ptr_array+m + 1 + j);
    i = 0;
    j = 0;
    k = l;
    while (i < n1 && j < n2)
    {
        if (L[i] <= R[j])
        {
            *(ptr_array+k) = L[i];
            i++;
        }
        else
        {
            *(ptr_array+k) = R[j];
            j++;
        }
        k++;
    }
    while (i < n1)
    {
        *(ptr_array+k) = L[i];
        i++;
        k++;
    }
    while (j < n2)
    {
        *(ptr_array+k) = R[j];
        j++;
        k++;
    }
}

void mergeSort(int * ptr_array, int l, int r)
{
    if (l < r)
    {
        int m = l+(r-l)/2;
        mergeSort(ptr_array, l, m);
        mergeSort(ptr_array, m+1, r);
        merge(ptr_array, l, m, r);
    }
}
```

**The Result of the following code is attached below:**

```
C:\Users\Hp\Documents\CodeBlocks\C\DataStructures\Lab08\bin\Debug\Lab08.exe
How do you want to sort the Array?
1-Bubble Sort?
2-Selection Sort?
3-Insertion Sort?
4-Merge Sort?

Enter your choice? 1-4
4

The Array sorted by Merge sort is:

 1      2      3      4      7      8      9      19      29      40      71      73      108      108      110
 130     134     138     151     156     159     159     167     167     179     182     190     202     206
 207     213     213     231     233     242     243     246     255     269     273     283     284     288
 295     296     301     301     318     327     331     334     339     340     348     385     391
 397     397     410     418     419     446     450     453     461     472     474     474     477     482
 497     508     516     533     537     560     562     576     577     581     581     633     636     649
 658     671     681     684     688     692     700     704     711     714     715     716     719     720
 722     722     727     736     740     755     757     776     777     780     788     790     794     813
 854     859     859     869     900     900     904     929     934     938     939     947     977     990
 999

Time elapsed is 1.535000 seconds
Process returned 0 (0x0)   execution time : 1.593 s
Press any key to continue.
```

The time of merge sort is already mentioned in task 2 table, though I will mention it here again,

Data Size	Merge Sort
16	0.74s
128	0.47s
1024	0.64s
16384	1.78s
131072	11.7s

-----  
**THE END**