Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **BS(CS)-4A**

LAB-JOURNAL-11

Exercise 1:

Implement the following sorting algorithms using a separate function for each.

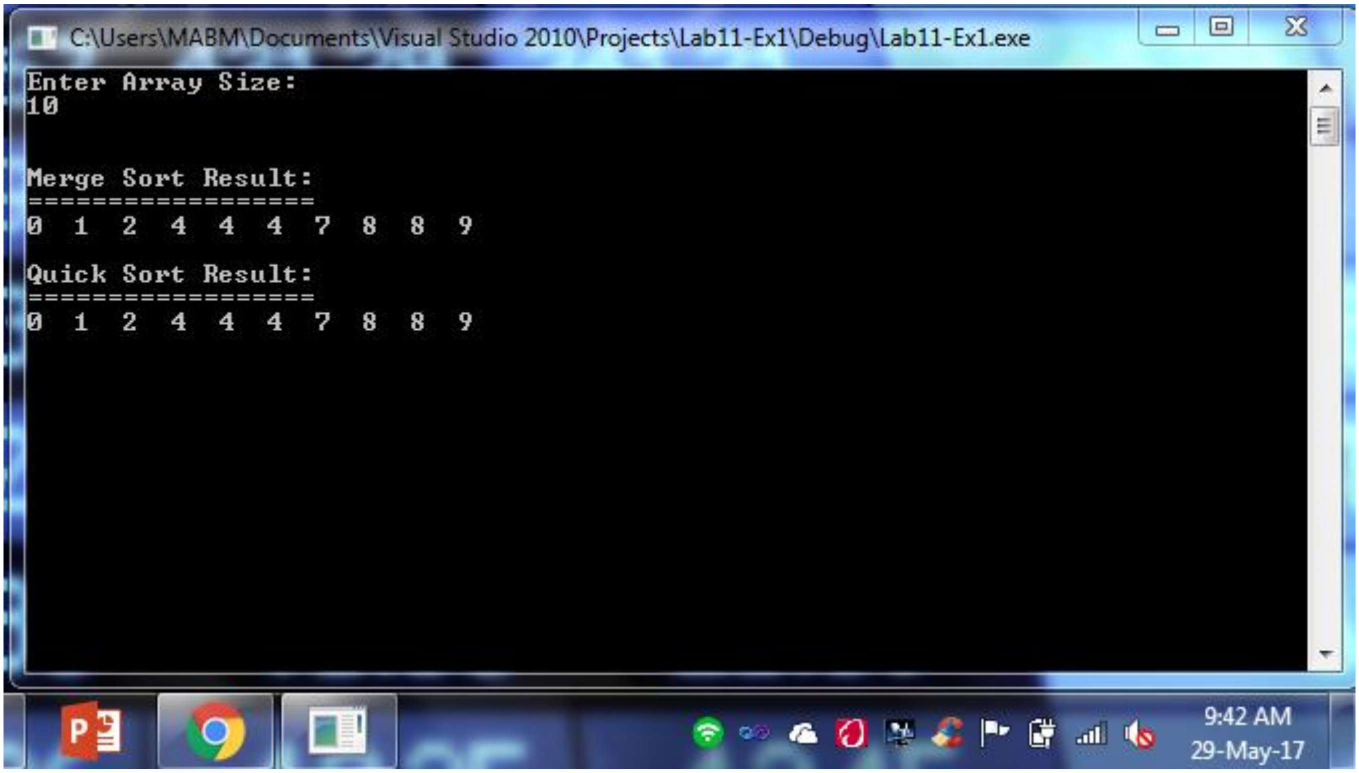
- Merge Sort
- Quick Sort

Solution:

main.cpp File:

```cpp
#include "conio.h"
#include "ctime"
#include <iostream>
using namespace std;

void merge(int *a, int *b, int low, int pivot, int high)
{
    int h, i, j, k;
    h = low;
    i = low;
    j = (pivot+1);

    while( ( h<=pivot ) && ( j<=high ) )
    {
        if(a[h]<=a[j])
        {
            b[i] = a[h];
            h++;
        }
        else
        {
            b[i]=a[j];
            j++;
        }
        i++;
    }

    if(h>pivot)
    {
        for(k=j; k<=high; k++)
        {
            b[i]=a[k];
            i++;
        }
    }
    else
    {
        for(k=h; k<=pivot; k++)
        {
            b[i]=a[k];
            i++;
        }
    }

    for(k=low; k<=high; k++)
    {
            a[k]=b[k];
    }
}

void mergeSort(int *a, int*b, int low, int high)
{
    int pivot;
    if( low<high )
    {
        pivot=( ( low+high )/2 );
        mergeSort( a, b, low, pivot);
        mergeSort(a, b, (pivot+1), high);
        merge(a, b, low, pivot, high);
    }
}

void split( int x[], int first, int last, int &pos )
{
    int pivot = x[first];
    int left = first;
    int right = last;
    while (left < right)
    {
        while( x[right] > pivot)
        {
            right--;
        }
        while( x[left] <= pivot && left < right )
        {
            left++;
        }
        if (left < right)
        {
            int temp;
            temp = x[left];
            x[left] = x[right];
            x[right] = temp;
        }
    }
        x[first] = x[right];
        x[right] = pivot;
        pos = right;
}

void quickSort (int x[], int first, int last)
{
    int pos;
    if ( first < last-1)
    {
        split (x, first, last, pos);
        quickSort (x, first, (pos-1));
        quickSort (x, pos + 1, last);
    }
      }

      int main()
      {
          int n;
          cout<<"Enter Array Size:"<<endl;
          cin>>n;
          //NOTE: Below individual arrays are declred for each sorting algorithm and then initialized with same elements to keep consistency in comparing algorithms working with same elements of array
          int *d = new int [n]; //array for merge sorting
          int *e = new int [n]; //array for quick sorting
          int *x = new int [n]; //array for merge sorting
          for(int i=0; i<n; i++) //initializing each array with same random numbers
          {
              d[i] = e[i] = (rand() % n);
          }

          //========================================================================================
          //merge sorting
          //========================================================================================
          cout<<endl;
          cout<<endl;
          cout<<"Merge Sort Result:"<<endl;
          cout<<"=================="<<endl;
          mergeSort(d, x, 0, n-1);
          for( int i=0; i<n; i++ )
          {
              cout<<d[i]<<"  ";
          }

          //========================================================================================
          //quick sorting
          //========================================================================================
          cout<<endl;
          cout<<endl;
          cout<<"Quick Sort Result:"<<endl;
          cout<<"=================="<<endl;
          quickSort(e, 0, n-1);
          for( int i=0; i<n; i++ )
          {
              cout<<e[i]<<"  ";
          }

          getch();
      }
```

Output:


C:\Users\MABM\Documents\Visual Studio 2010\Projects\Lab11-Ex1\Debug\Lab11-Ex1.exe
9:42 AM
29-May-17

```
Enter Array Size:
10

Merge Sort Result:
==================
0  1  2  4  4  4  7  8  8  9

Quick Sort Result:
==================
0  1  2  4  4  4  7  8  8  9
```

Exercise 2:

Generate a random list of 1,000 elements in the range [0 999]. Using shell sort algorithm, find the total number of comparisons/array element shifts carried out for the given set of span (number of sub-files) values.

- 25, 10, 5, 1
- 100, 50, 25, 10, 1
- 5, 3, 1

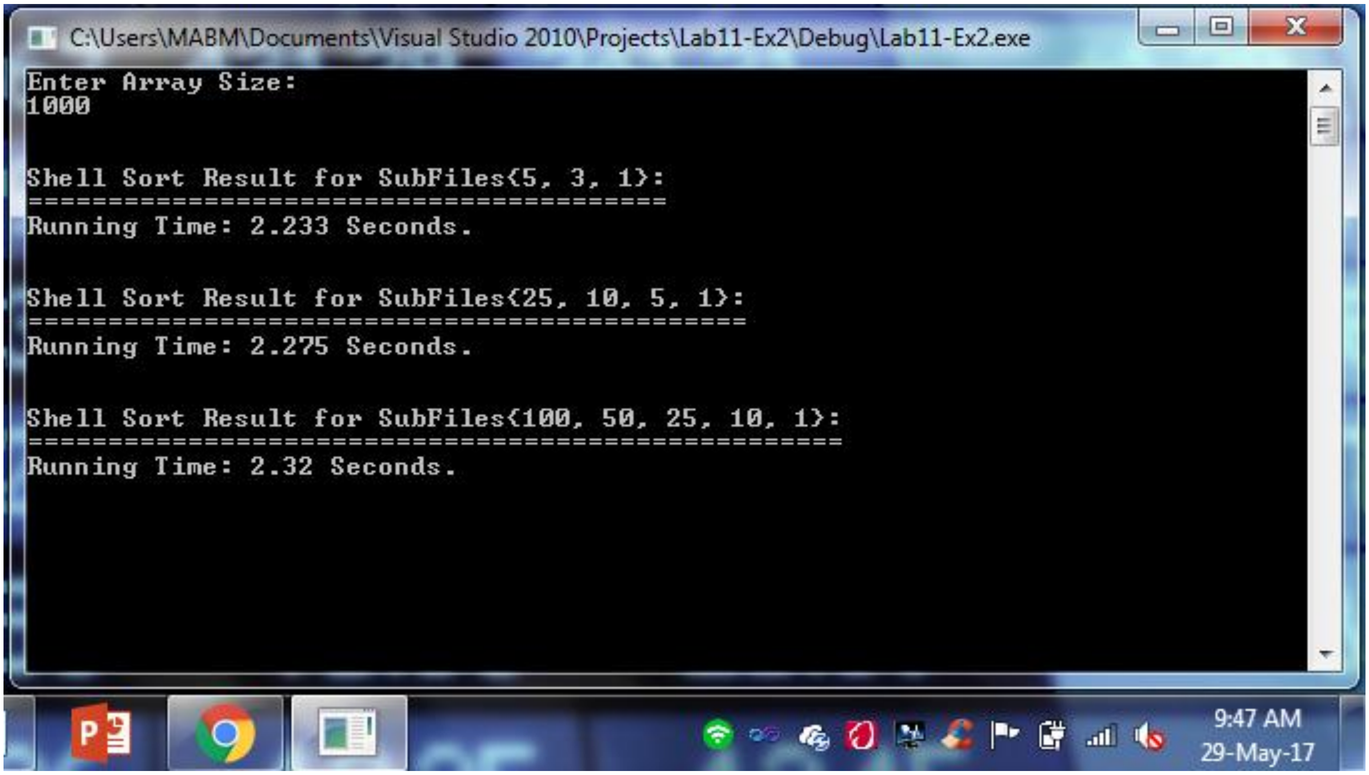
Also compute the execution times of each of the above scenarios.

Solution:

main.cpp File:

```
#include "conio.h"
#include "ctime"
#include "ctime"
#include <iostream>
using namespace std;

void shellSort(int x[], int n, int incrmnts[], int numinc)
{
    int span, y;

    for(int incr = 0; incr < numinc; incr++)
    {
        span = incrmnts[incr]; //span is the size of increment

        for(int j = span; j<n; j++)
        {
            y = x[j]; //insert x[j] at its proper location within its subfile using simple insert sort
            int k;
            for(k = j-span;k>=0 && y<x[k]; k-=span)
            {
                    x[k+span] = x[k];
            }
            x[k+span] = y;
        }
    }
}

int main()
{
    int n;
    cout<<"Enter Array Size:"<<endl;
    cin>>n;

    int *a = new int [n]; //array for shell sorting for 3 subfiles
    int *b = new int [n]; //array for shell sorting for 4 subfiles
    int *c = new int [n]; //array for shell sorting for 5 subfiles

    int x1[3] = {5, 3, 1}; //array for shell sorting for 3 subfiles
    int x2[4] = {25, 10, 5, 1}; //array for shell sorting for 4 subfiles
    int x3[5] = {100, 50, 25, 10, 1}; //array for shell sorting for 5 subfiles

    for(int i=0; i<n; i++) //initializing each array with same random numbers
    {
        a[i] = b[i] = c[i] = (rand() % n);
    }

    //=======================================================================================
    //shell sorting
    //=======================================================================================
    cout<<endl;
    cout<<endl;
    cout<<"Shell Sort Result for SubFiles{5, 3, 1}:"<<endl;
    cout<<"========================================"<<endl;
    clock_t a_time;
    a_time = clock();
    shellSort(a, n, x1, 3);
    cout<<"Running Time: "<<(float)a_time/CLOCKS_PER_SEC<<" Seconds."<<endl;

    cout<<endl;
    cout<<endl;
    cout<<"Shell Sort Result for SubFiles{25, 10, 5, 1}:"<<endl;
    cout<<"============================================="<<endl;
    clock_t b_time;
    b_time = clock();
    shellSort(b, n, x2, 4);
    cout<<"Running Time: "<<(float)b_time/CLOCKS_PER_SEC<<" Seconds."<<endl;

    cout<<endl;
    cout<<endl;
    cout<<"Shell Sort Result for SubFiles{100, 50, 25, 10, 1}:"<<endl;
    cout<<"==================================================="<<endl;
    clock_t c_time;
    c_time = clock();
    shellSort(c, n, x3, 5);
    cout<<"Running Time: "<<(float)c_time/CLOCKS_PER_SEC<<" Seconds."<<endl;

    getch();
}
```

Output:



C:\Users\MABM\Documents\Visual Studio 2010\Projects\Lab11-Ex2\Debug\Lab11-Ex2.exe
9:47 AM
29-May-17

```
Enter Array Size:
1000

Shell Sort Result for SubFiles(5, 3, 1):
========================================
Running Time: 2.233 Seconds.

Shell Sort Result for SubFiles(25, 10, 5, 1):
=============================================
Running Time: 2.275 Seconds.

Shell Sort Result for SubFiles(100, 50, 25, 10, 1):
===================================================
Running Time: 2.32 Seconds.
```