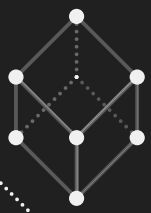
UNIVERSITY OF
ENGINEERING AND TECHNOLOGY
MARDAN



Name

Abdul Haseeb

Reg. No

22MDSWE197

Class. No

4

DSA Lab.

4

Submitted To.

Engr. Sohail

1. Implement bubble sort as a function which takes a float array and the size of the array as parameters and returns the array sorted in descending order.

2. Implement Insertion sort as a function which takes a float array and the size of the array as parameters and returns the array sorted in descending order

3. Create a function named “InitArray” which takes an integer parameter N for the number of elements in the array and creates a float array of N elements randomly initializing the array with numbers between 0 and 1000. (Note: use rand() function).

4. Call both functions created in task 1 and task 2 from the main function with the float array in task 3 as parameter to both the functions. Display the sorted array returned by each function.

5. By adding the execution time calculation logic used in the Assignment # 1, Calculate the execution time for both algorithms and fill the values in the following table.
N=10 N=100 N=1000
Bubble Sort Execution Time
Insertion Sort Execution Time

6. Submit a printed report of the code snapshots (full screen) and output snapshots

1. Implement bubble sort as a function which takes a float array and the size of the array as parameters and returns the array sorted in descending order.

Bubble Sort

```cpp

#include<iostream>

using namespace std;

void bubbleSort(int array[], int len){

    int temp;

    for(int i=0; i<len; i++){

        for(int j=0; j<5-i-1; j++){

            if(array[j] < array[j+1]){

                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }

    }

}



int main(){

    cout<<"Please enter the size of the array: " ;
    int size;
    cin>> size;

    int array [size];
    cout<<"\n Now enter the elements of the array: "<<endl;

    for(int i=0; i<size; i++){
         cin>>array[i];

    }

bubbleSort(array, size);

        cout<<"The sorted array is: "<<endl;

    for(int i=0; i<size; i++){

        cout<<"\n"<<array[i];

    }
}


```

Output

```
Please enter the size of the array: 7

 Now enter the elements of the array:
4
2
7
1
9
3
6
The sorted array is:

9
7
4
2
1
3
6
```

2. Implement Insertion sort as a function which takes a float array and the size of the array as parameters and returns the array sorted in descending order

Insertion Sort

```cpp
#include<iostream>

using namespace std;



void insertionSort(int arr[], int len){

     int i,j,temp;

    for(int i=1; i<len; i++){

        j=i;
        temp = arr[i];

        while(j>0 && temp> arr[j-1]){

            arr[j] = arr[j -1];

            j--;

        }

        arr[j] = temp;

    }

}



int main(){


    cout<<"Please enter the size of the array: " ;
    int size;
    cin>> size;

    int array [size];
    cout<<"\n Now enter the elements of the array: "<<endl;

    for(int i=0; i<size; i++){
         cin>>array[i];

    }


    insertionSort(array, size);



    cout<<"after sorting the array: "<<endl;

    for(int i=0; i<size; i++){
         cout<<array[i];

    }
}
```

Output

```
nsertionSort }
Please enter the size of the array: 8

 Now enter the elements of the array:
7
3
7
1
5
7
4
5
after sorting the array:
77755431
```

3. Create a function named “InitArray” which takes an integer parameter N for the number of elements in the array and creates a float array of N elements randomly initializing the array with numbers between 0 and 1000. (Note: use rand() function).

initArray

```cpp
#include<iostream>
#include <cstdlib>  // Required for rand() function
#include <ctime>
using namespace std;




void InitArray(float arr[], int N) {


    for (int i = 0; i < N; i++) {

        float randomFloat = (rand()%1000 ) *(0.9 - rand()%1);

        arr[i] = randomFloat;
    }
}



int main(){


cout<<"Enter the number of random floating points elements: ";

int size;
cin>> size;

float array[size];

InitArray(array, size);


for(int i=0; i<size; i++){
    cout<<array[i]<<endl;
}




}
```

Output

```
.\3RandomGenerator }
Enter the number of random floating points elements: 11
420.3
450
651.6
322.2
417.6
130.5
744.3
441.9
847.8
392.4
543.6
```

4. Call both functions created in task 1 and task 2 from the main function with the float array in task 3 as parameter to both the functions. Display the sorted array returned by each function.

Float Sorting Algorithms

```c
void bubbleSort(float array[], int len){

    float temp;

    for(int i=0; i<len-1; i++){



        for(int j=0; j<len-i; j++){

            if(array[j] < array[j+1]){

                temp = array[j];
                array[j] = array[j+1];
                array[j+1] = temp;
            }
        }

    }

}
```

```c
void insertionSort(float arr[], float len){

     int i,j,temp;

    for(int i=1; i<len; i++){

        j=i;
        temp = arr[i];

        while(j>0 && temp> arr[j-1]){

            arr[j] = arr[j -1];

            j--;

        }

        arr[j] = temp;

    }
}
```

Output [main() is on next page]

```
Please enter the size of the array: 6

 Now enter the float elements of the array:
21
43
12
453
12
43
Sorted using bubble sort:
453
43
43
21
12
12

address of array0x7bdfdff6e0


Please enter the size of the array: 4

 Now enter the float elements of the array:
12
43
12
65
Sorted using insertion sort:
65
43
12
12

address of array0x7bdfdff6e0
```

Main.cpp

```cpp
#include "sortLibraries/bubble.h"
#include "sortLibraries/insertion.h"
#include<iostream>

using namespace std;

int main(){

    cout<<"Please enter the size of the array: " ;
    int size;
    cin>> size;

    float array [size];
    cout<<"\n Now enter the float elements of the array: "<<endl;

    for(int i=0; i<size; i++){
         cin>>array[i];

    }

    bubbleSort(array, size);

    cout<<"Sorted using bubble sort: \n";

    for(int i=0; i<size; i++){
        cout<<array[i]<<endl;
    }

    cout<<"\naddress of array"<<array<<"\n\n";


   cout<<"\n\nPlease enter the size of the array: " ;
     size;
    cin>> size;

     array [size];
    cout<<"\n Now enter the float elements of the array: "<<endl;


    for(int i=0; i<size; i++){
         cin>>array[i];

    }


    cout<<"Sorted using insertion sort: \n";

  insertionSort(array, size);

  for(int i=0; i<size; i++){
    cout<<array[i]<<endl;
  }

cout<<"\naddress of array"<<array<<"\n";


}
```

5. By adding the execution time calculation logic used in the Assignment # 1, Calculate the execution time for both algorithms and fill the values in the following table.
N=10 N=100 N=1000
Bubble Sort Execution Time
Insertion Sort Execution Time

```cpp

#include<iostream>
#include "sortLibraries/bubble.h"
#include "sortLibraries/insertion.h"
#include "sortLibraries/randomGenerator.h"
#include<chrono>
using namespace std;
using namespace chrono;
int main(){


float array [1000] ;

InitArray(array, 1000);


auto startTime = high_resolution_clock::now();

bubbleSort(array, 1000);

auto endTime = high_resolution_clock::now();

auto differenceTime = duration_cast<milliseconds>(endTime - startTime);

cout<<"Duration is "<<differenceTime.count();

cout<<"\n";


InitArray(array, 1000);


auto startTime = high_resolution_clock::now();

insertionSort(array, 1000);

auto endTime = high_resolution_clock::now();

auto differenceTime = duration_cast<milliseconds>(endTime - startTime);

cout<<"Duration is "<<differenceTime.count();

// cout<<"\n";
// for(int i=0; i<1000; i++){
//     cout<<array[i]<< " ";
// }


    return 0;
}
```

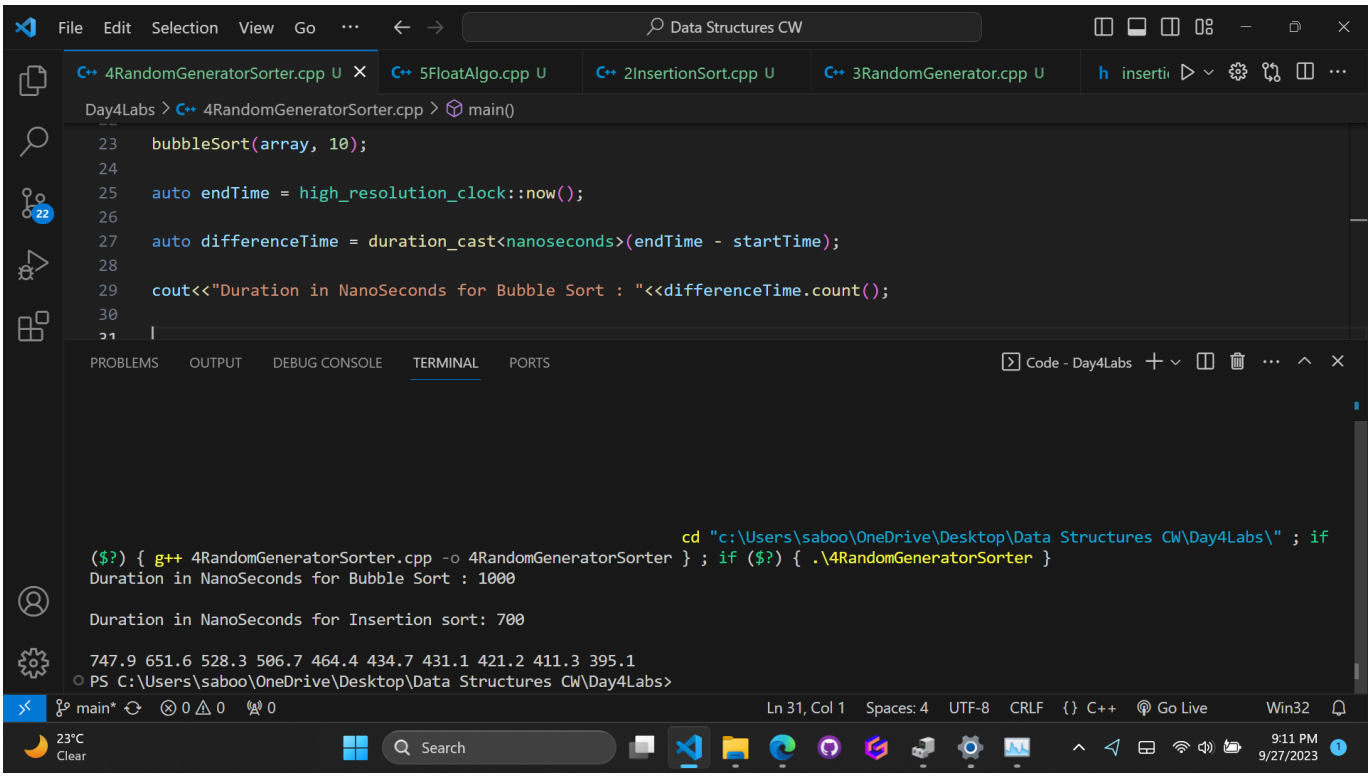
5. By adding the execution time calculation logic used in the Assignment # 1, Calculate the execution time for both algorithms and fill the values in the following table.
N=10 N=100 N=1000
Bubble Sort Execution Time
Insertion Sort Execution Time

```cpp

#include<iostream>
#include "sortLibraries/bubble.h"
#include "sortLibraries/insertion.h"
#include "sortLibraries/randomGenerator.h"
#include<chrono>
using namespace std;
using namespace chrono;
int main(){


float array [1000] ;

InitArray(array, 1000);


auto startTime = high_resolution_clock::now();

bubbleSort(array, 1000);

auto endTime = high_resolution_clock::now();

auto differenceTime = duration_cast<milliseconds>(endTime - startTime);

cout<<"Duration is "<<differenceTime.count();

cout<<"\n";


InitArray(array, 1000);


auto startTime = high_resolution_clock::now();

insertionSort(array, 1000);

auto endTime = high_resolution_clock::now();

auto differenceTime = duration_cast<milliseconds>(endTime - startTime);

cout<<"Duration is "<<differenceTime.count();

// cout<<"\n";
// for(int i=0; i<1000; i++){
//     cout<<array[i]<< " ";
// }


    return 0;
}
```

6. Submit a printed report of the code snapshots (full screen) and output snapshots

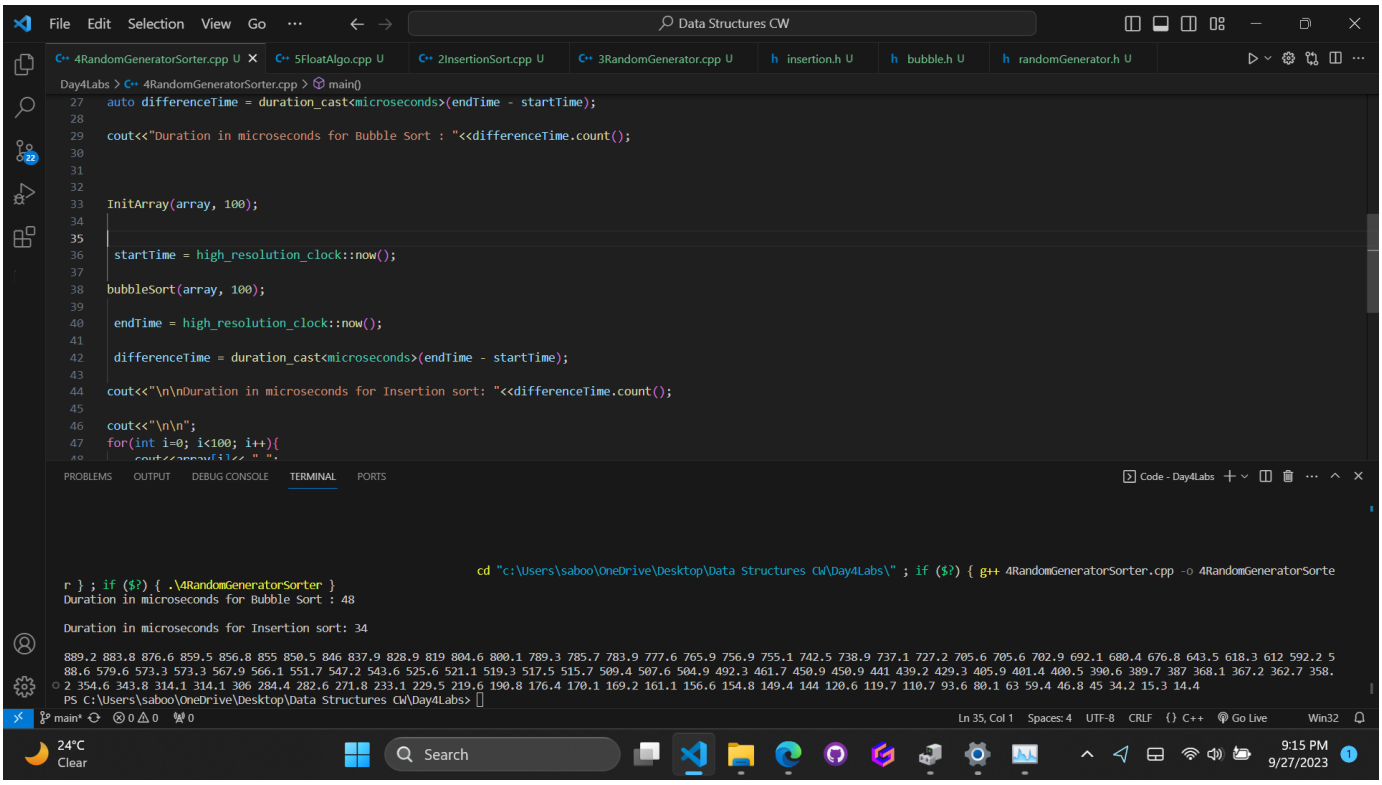
For N=10

Duration in NanoSeconds for Bubble Sort : 1000
Duration in NanoSeconds for Insertion sort: 700



For N=100

Duration in microseconds for Bubble Sort : 48
Duration in microseconds for Insertion sort: 34

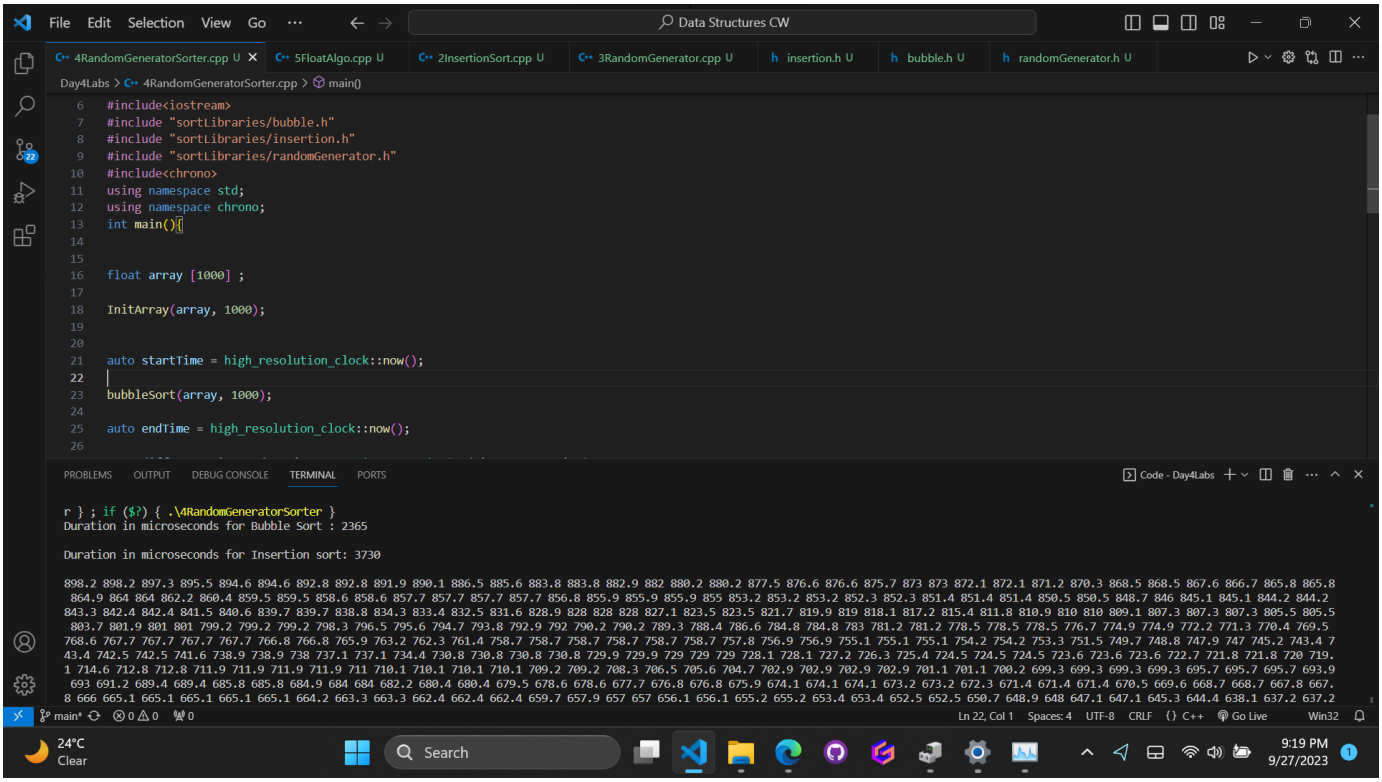


6. Submit a printed report of the code snapshots (full screen) and output snapshots

For N=1000

Duration in microseconds for Bubble Sort : 2365

Duration in microseconds for Insertion sort: 3730



The End.

Submitted to : Engr. Sohail

Date: 27 / 9 / 2023

22MDSWE197

04 Abdul Haseeb
