LAB REPORT

CSE 114 : Data Structure and Algorithms Sessional

PREPARED BY

Mehrin Farzana
ID: 2101013
Session: 2021-2022
Date: 22/07/2023

SUPERVISED BY

Suman Saha
Lecturer
Department of IRE, BDU

BANGABANDHU SHEIKH MUJIBUR

RAHMAN DIGITAL UNIVERSITY

(BDU)

List of Problems

1. Write down a program that implements three sorting algorithms (bubble sort, selection sort, and insertion sort). Using a switch statement to choose the desired algorithm.

2. Consider a random array of n different sizes. Now write down a program that measures and records the execution time for each sorting algorithm (bubble sort, selection sort, and insertion sort) to sort the generated arrays. Repeat the experiment multiple times for each input size and calculate the average execution time.

3. Implement optimized versions of the sorting algorithms (bubble sort, selection sort, and insertion sort) to improve the performance.

Problem No.: 01

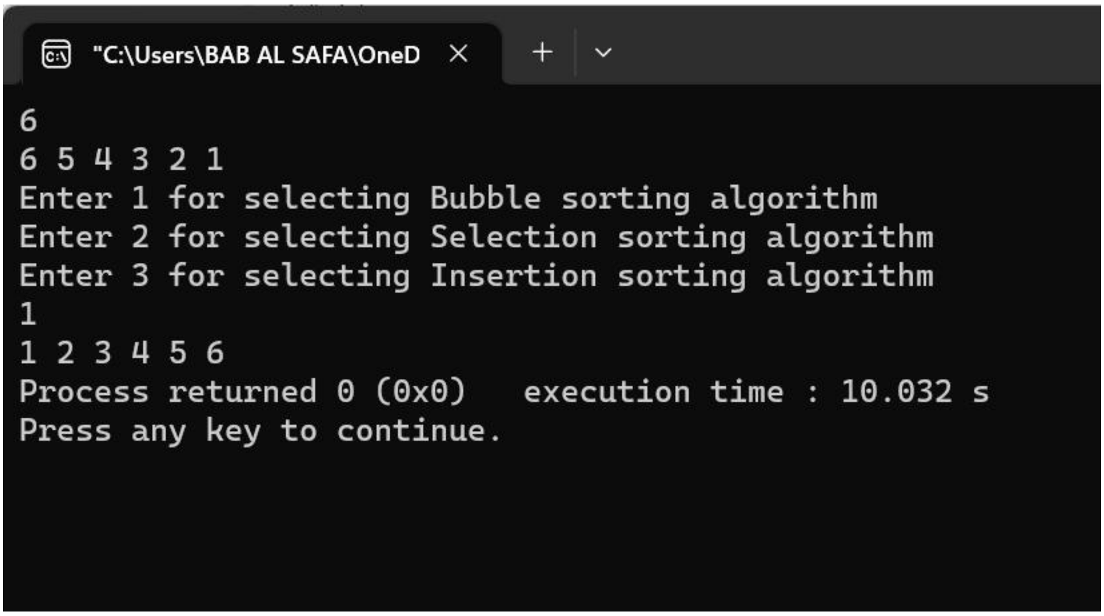
Problem Statement:

Write down a program that implements three sorting algorithms (bubble sort, selection sort, and insertion sort). Using a switch statement to choose the desired algorithm.

Code:

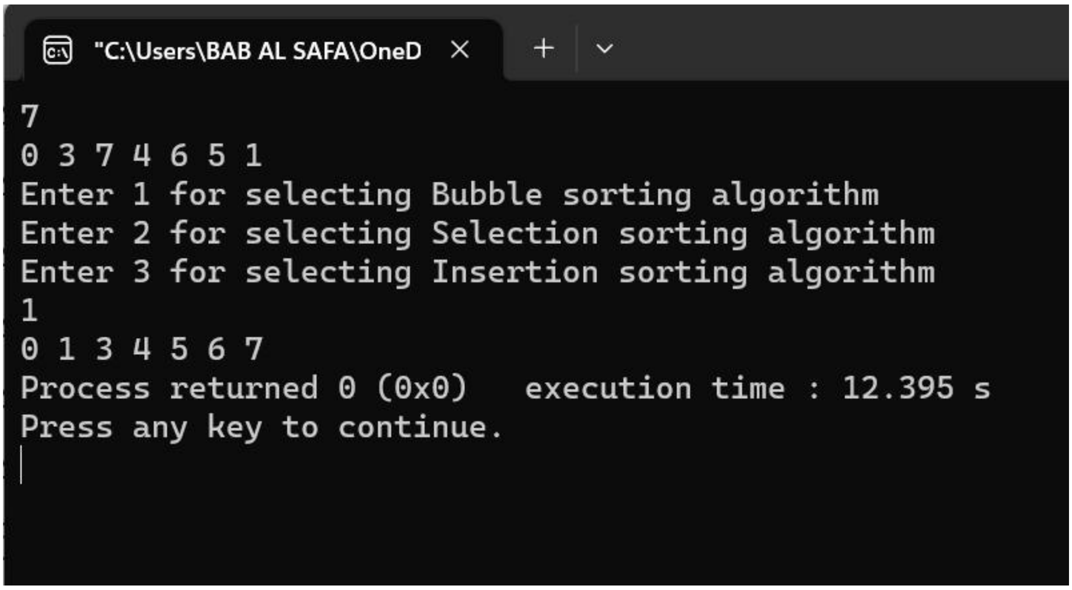
```
#include <stdio.h>
void bubble_sort(int *a, int n){
    for(int i=0; i<n-1; i++){
        for(int j=0; j<n-1-i; j++){
            if(a[j]>a[j+1]){
                int tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
    }
}
void selection_sort(int *a, int n){
    int min, min_pos;
    for(int j=0; j<n; j++){
            min = a[j];
        for(int i=j; i<n; i++){
            if(min>=a[i]){
                min = a[i];
                min_pos = i;

            }
        }
        int tmp = a[j];
        a[j] = a[min_pos];
        a[min_pos] = tmp;
    }
}
void insertion_sort(int *a, int n){
    int key, i, j;
    for(i=1; i<n; i++){
        key = a[i];
        for(j=i-1; j>=0 && key<a[j]; j--){
            a[j+1]=a[j];
        }
        a[j+1]=key;
    }
}
int main() {
    int n, choose;
    scanf("%d", &n);
    int a[n];
    for(int i=0; i<n; i++)
        scanf("%d", &a[i]);

    printf("Enter 1 for selecting Bubble sorting algorithm\nEnter 2 for selecting Selection sorting algorithm\nEnter 3 for selecting Insertion sorting algorithm\n");
    scanf("%d", &choose);
    switch(choose){
        case(1):
            bubble_sort(a,n);
            break;
        case(2):
            selection_sort(a,n);
            break;
        case(3):
            insertion_sort(a,n);
            break;
    }

    for(int i=0; i<n; i++)
        printf("%d ", a[i]);

    return 0;
}
```

Output:



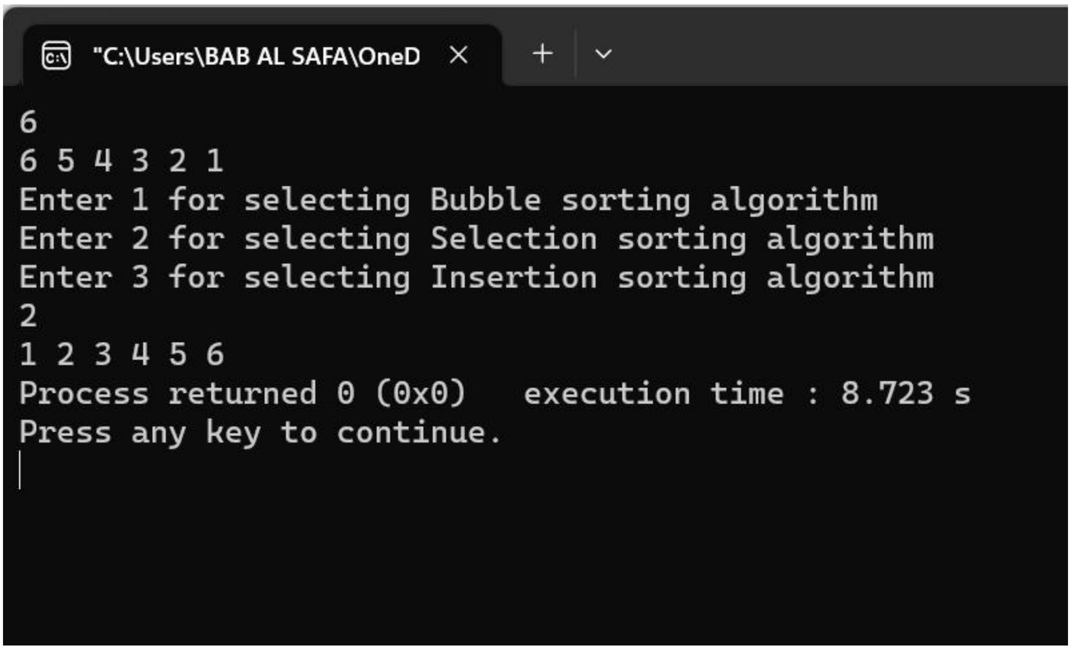
```
6
6 5 4 3 2 1
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
1
1 2 3 4 5 6
Process returned 0 (0x0)   execution time : 10.032 s
Press any key to continue.
```

Fig 1.1: Output on console for case 1.



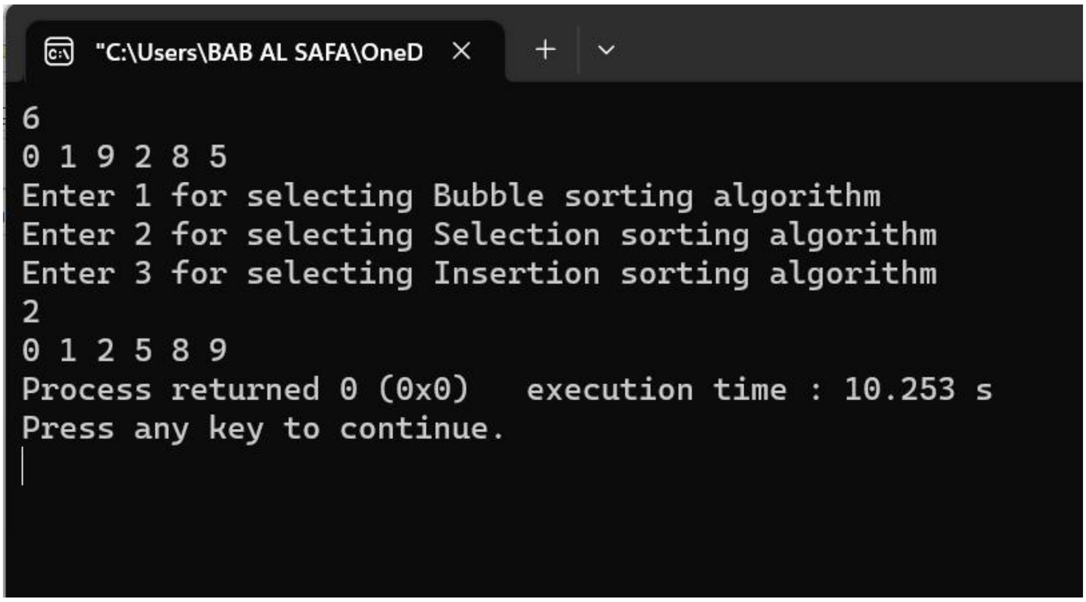
```
7
0 3 7 4 6 5 1
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
1
0 1 3 4 5 6 7
Process returned 0 (0x0)   execution time : 12.395 s
Press any key to continue.
```

Fig 1.2: Output on console for case 2.



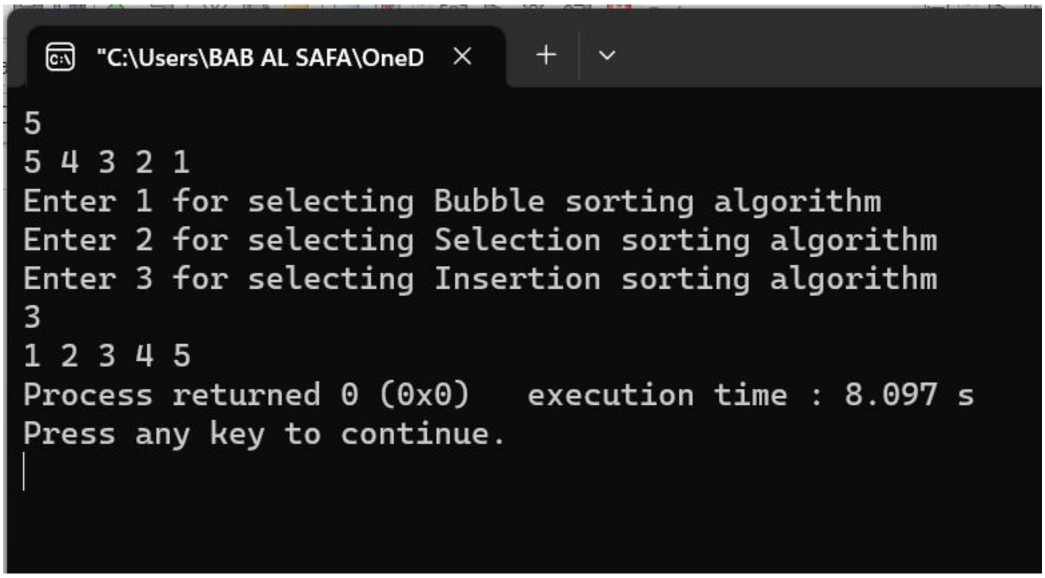
```
6
6 5 4 3 2 1
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
2
1 2 3 4 5 6
Process returned 0 (0x0)   execution time : 8.723 s
Press any key to continue.
```

Fig 1.3: Output on console for case 3.



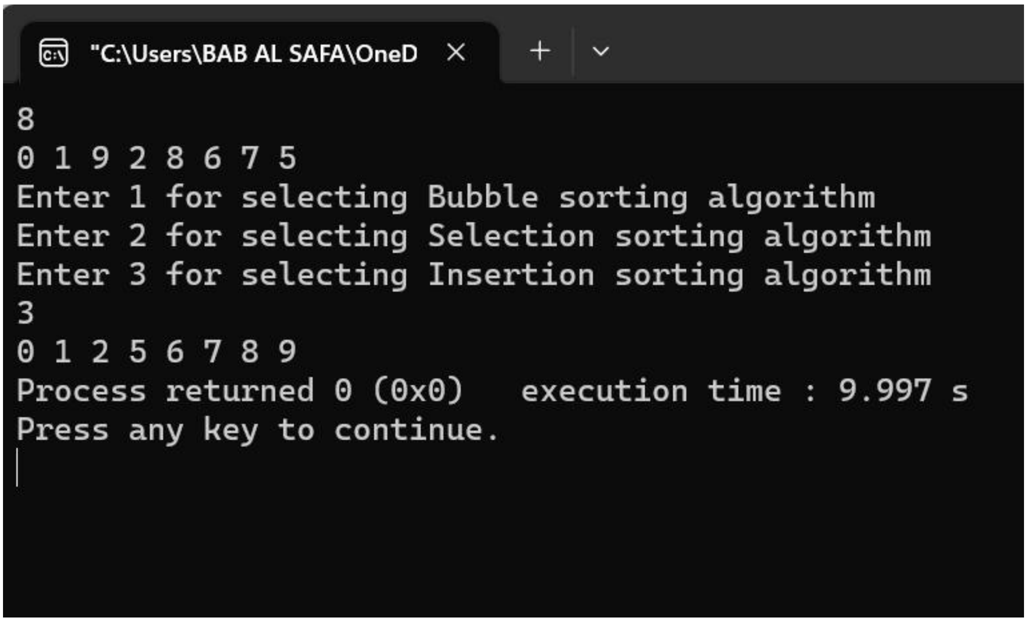
```
6
0 1 9 2 8 5
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
2
0 1 2 5 8 9
Process returned 0 (0x0)   execution time : 10.253 s
Press any key to continue.
```

Fig 1.4: Output on console for case 4.



```
5
5 4 3 2 1
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
3
1 2 3 4 5
Process returned 0 (0x0)   execution time : 8.097 s
Press any key to continue.
```

Fig 1.5: Output on console for case 5.



```
8
0 1 9 2 8 6 7 5
Enter 1 for selecting Bubble sorting algorithm
Enter 2 for selecting Selection sorting algorithm
Enter 3 for selecting Insertion sorting algorithm
3
0 1 2 5 6 7 8 9
Process returned 0 (0x0)   execution time : 9.997 s
Press any key to continue.
```

Fig 1.6: Output on console for case 6.

Problem No.: 02

Problem Statement:

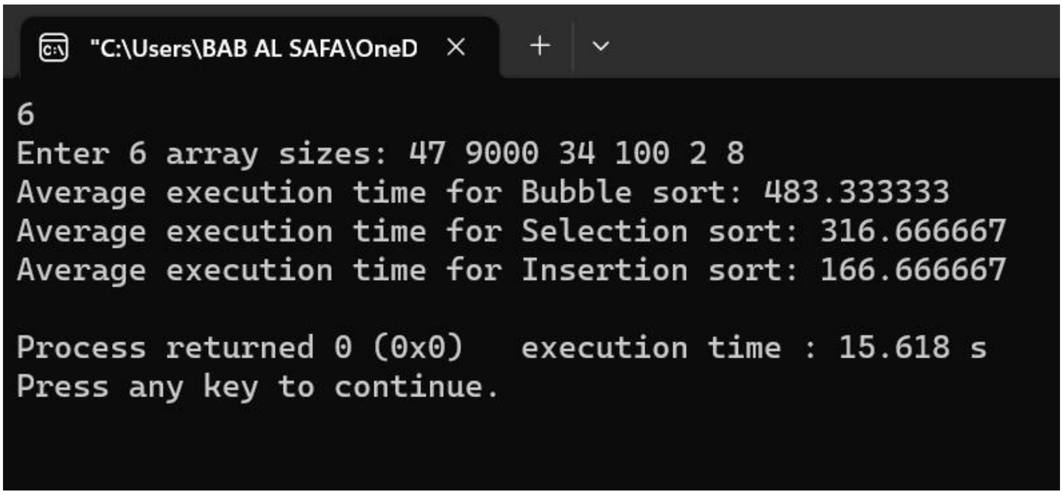
Consider a random array of n different sizes. Now write down a program that measures and records the execution time for each sorting algorithm (bubble sort, selection sort, and insertion sort) to sort the generated arrays. Repeat the experiment multiple times for each input size and calculate the average execution time.

Code:

```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
void bubble_sort(int *a, int n){
   for(int i=0; i<n-1; i++){
      for(int j=0; j<n-1-i; j++){
         if(a[j]>a[j+1]){
            int tmp = a[j];
            a[j] = a[j+1];
            a[j+1] = tmp;
         }
      }
   }
}
void selection_sort(int *a, int n){
   int min, min_pos;
   for(int j=0; j<n; j++){
         min = a[j];
      for(int i=j; i<n; i++){
         if(min>=a[i]){
            min = a[i];
            min_pos = i;

         }
      }
      int tmp = a[j];
      a[j] = a[min_pos];
      a[min_pos] = tmp;
   }
}
void insertion_sort(int *a, int n){
   int key, i, j;
   for(i=1; i<n; i++){
      key = a[i];
      for(j=i-1; j>=0 && key<a[j]; j--){
         a[j+1]=a[j];
      }
      a[j+1]=key;
```

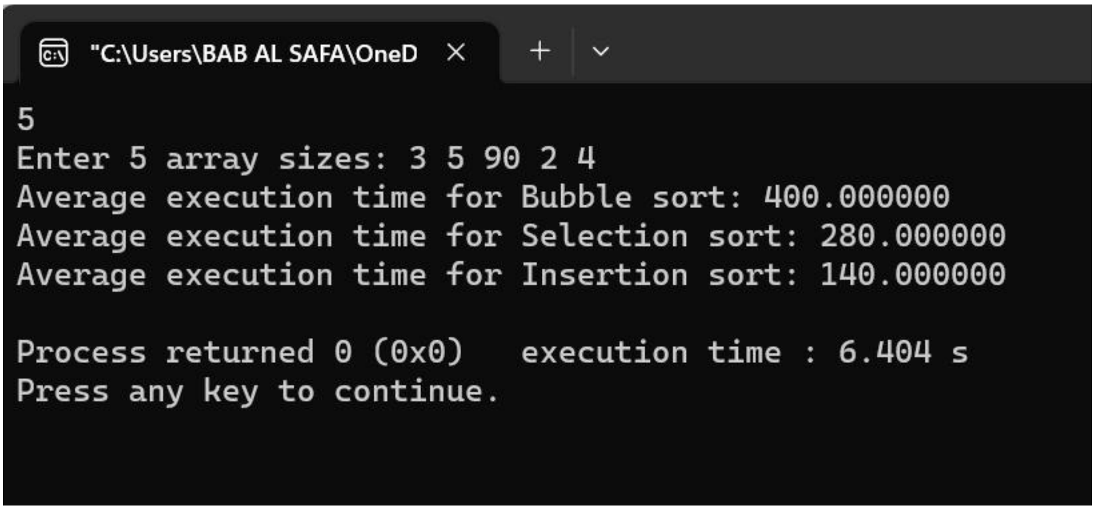
```
    }
}
int main() {
    srand(time(NULL));
    struct timespec start_time, end_time;
    int n, s;
    double sum_bubble=0, sum_selection=0, sum_insertion=0;
    scanf("%d", &n);
    printf("Enter %d array sizes: ", n);
    for(int i=0; i<n; i++){
        scanf("%d", &s);
        int a[s];
        for(int j=0; j<s; j++){
            a[j] = rand()%100;
        }

        clock_gettime(CLOCK_MONOTONIC, &start_time);
        bubble_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        double elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 +
(end_time.tv_nsec - start_time.tv_nsec);
        sum_bubble+=elapsed_time;


        clock_gettime(CLOCK_MONOTONIC, &start_time);
        selection_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 + (end_time.tv_nsec
- start_time.tv_nsec);
        sum_selection+=elapsed_time;


        clock_gettime(CLOCK_MONOTONIC, &start_time);
        insertion_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 + (end_time.tv_nsec
- start_time.tv_nsec);
        sum_insertion+=elapsed_time;

        }


        printf("Average execution time for Bubble sort: %f\nAverage execution time for
Selection sort: %f\nAverage execution time for Insertion sort: %f\n", sum_bubble/n,
sum_selection/n, sum_insertion/n);


    return 0;
}
```

Output:



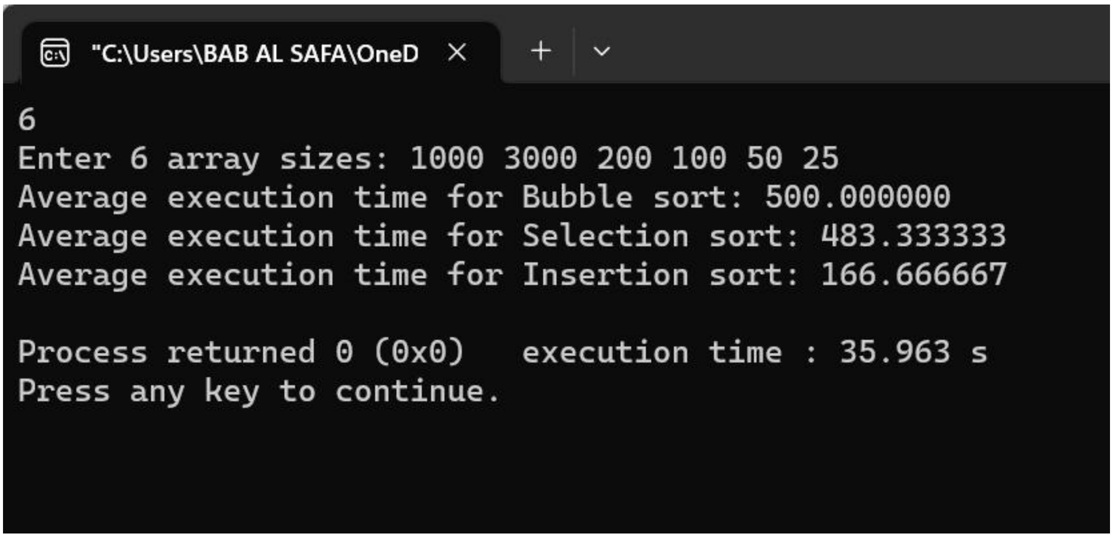
```
6
Enter 6 array sizes: 47 9000 34 100 2 8
Average execution time for Bubble sort: 483.333333
Average execution time for Selection sort: 316.666667
Average execution time for Insertion sort: 166.666667

Process returned 0 (0x0)   execution time : 15.618 s
Press any key to continue.
```

Fig 2.1: Output on console for case 1.



```
5
Enter 5 array sizes: 3 5 90 2 4
Average execution time for Bubble sort: 400.000000
Average execution time for Selection sort: 280.000000
Average execution time for Insertion sort: 140.000000

Process returned 0 (0x0)   execution time : 6.404 s
Press any key to continue.
```

Fig 2.2: Output on console for case 2.



```
6
Enter 6 array sizes: 1000 3000 200 100 50 25
Average execution time for Bubble sort: 500.000000
Average execution time for Selection sort: 483.333333
Average execution time for Insertion sort: 166.666667

Process returned 0 (0x0)   execution time : 35.963 s
Press any key to continue.
```

Fig 2.3: Output on console for case 3.

Problem No.: 03

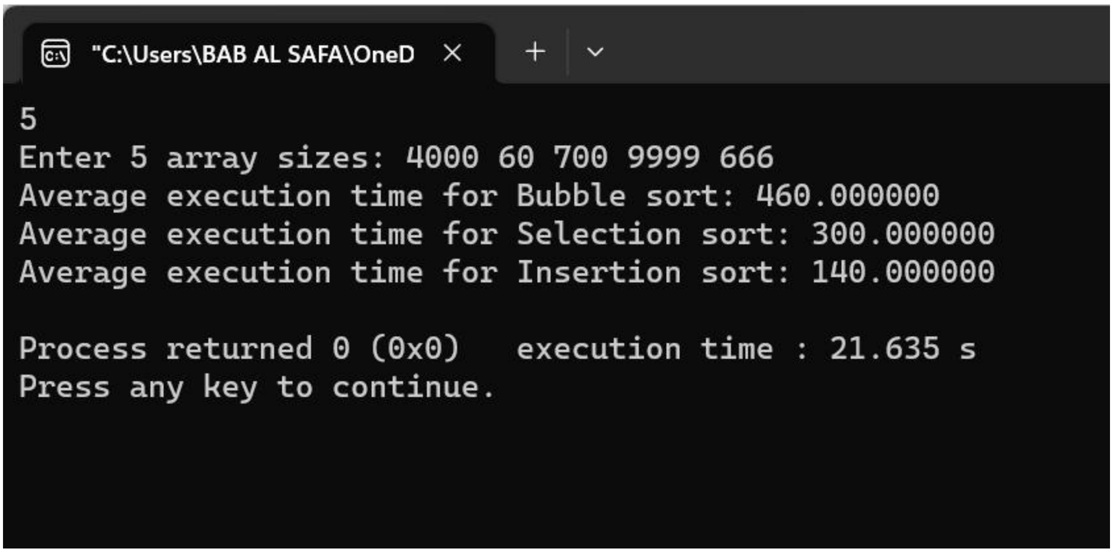
Problem Statement:

Implement optimized versions of the sorting algorithms (bubble sort, selection sort, and insertion sort) to improve the performance.

Code:

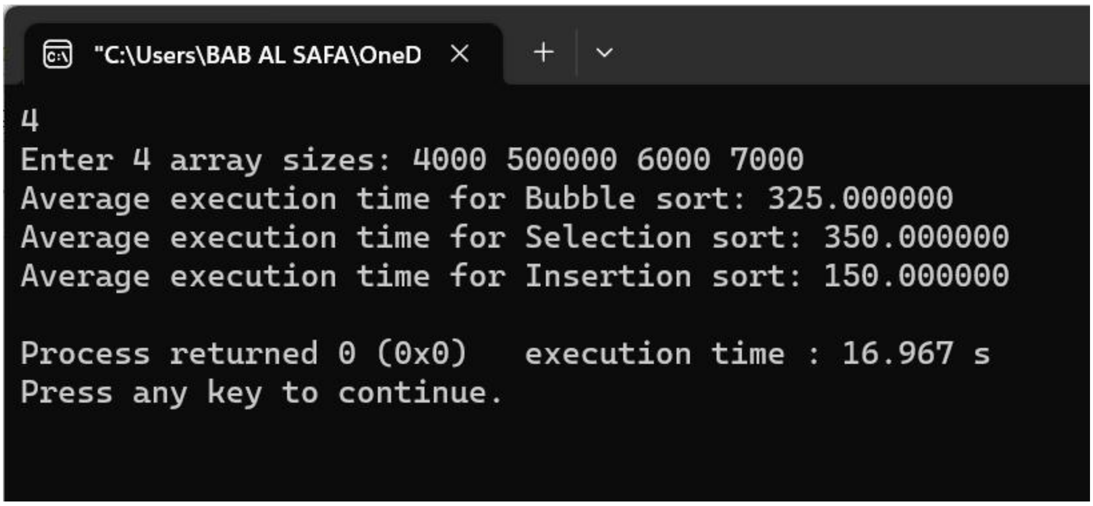
```
#include <stdio.h>
#include <time.h>
#include <stdlib.h>
void bubble_sort(int *a, int n){
    int flag = 0;
    for(int i=0; i<n-1; i++){
        flag=0;
        for(int j=0; j<n-1-i; j++){
            if(a[j]>a[j+1]){
                flag=1;
                int tmp = a[j];
                a[j] = a[j+1];
                a[j+1] = tmp;
            }
        }
        if(!flag){
            break;
        }
    }
}

void selection_sort(int *a, int n){
    int min, min_pos,max,max_pos;
    for(int j=0; j<n-1-j; j++){
            min = a[j];
            max = a[j];
        for(int i=j; i<n-j; i++){
            if(min>a[i]){
                min = a[i];
                min_pos = i;

            }
            if(max<a[i]){
                max=a[i];
                max_pos=i;
            }
        }
        int tmp = a[j];
        a[j] = a[min_pos];
        a[min_pos] = tmp;

        if(a[min_pos]==max){
            tmp = a[n-j-1];
            a[n-j-1] = a[min_pos];
            a[min_pos] = tmp;
        }

        else{
            tmp = a[n-j-1];
            a[n-j-1] = a[max_pos];
            a[max_pos] = tmp;
        }
    }
}
void insertion_sort(int *a, int n){
    int key, i, j;
    for(i=1; i<n; i++){
        key = a[i];
        for(j=i-1; j>=0 && key<a[j]; j--){
            a[j+1]=a[j];
        }
        a[j+1]=key;
    }
}
int main() {
    int n, s, sum_bubble=0, sum_selection=0, sum_insertion=0;
    srand(time(NULL));
    struct timespec start_time, end_time;
    scanf("%d", &n);
    printf("Enter %d array sizes: ", n);
    for(int i=0; i<n; i++){
        scanf("%d", &s);
        int a[s];
        for(int j=0; j<s; j++){
            a[j] = rand()%100;
        }

        clock_gettime(CLOCK_MONOTONIC, &start_time);
        bubble_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        double elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 +
(end_time.tv_nsec - start_time.tv_nsec);
        sum_bubble+=elapsed_time;
        clock_gettime(CLOCK_MONOTONIC, &start_time);
        selection_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 + (end_time.tv_nsec - start_time.tv_nsec);
        sum_selection+=elapsed_time;


        clock_gettime(CLOCK_MONOTONIC, &start_time);
        insertion_sort(a,n);
        clock_gettime(CLOCK_MONOTONIC, &end_time);
        elapsed_time = (end_time.tv_sec - start_time.tv_sec) * 1e9 + (end_time.tv_nsec - start_time.tv_nsec);
        sum_insertion+=elapsed_time;

        }

        printf("Average execution time for Bubble sort: %f\nAverage execution time for Selection sort: %f\nAverage execution time for Insertion sort: %f\n", sum_bubble/n, sum_selection/n, sum_insertion/n);


    return 0;
}
```

Output:



```
5
Enter 5 array sizes: 4000 60 700 9999 666
Average execution time for Bubble sort: 460.000000
Average execution time for Selection sort: 300.000000
Average execution time for Insertion sort: 140.000000

Process returned 0 (0x0)   execution time : 21.635 s
Press any key to continue.
```

Fig 3.1: Output on console for case 1.



```
4
Enter 4 array sizes: 4000 500000 6000 7000
Average execution time for Bubble sort: 325.000000
Average execution time for Selection sort: 350.000000
Average execution time for Insertion sort: 150.000000

Process returned 0 (0x0)   execution time : 16.967 s
Press any key to continue.
```

Fig 3.2: Output on console for case 2.