Green University of Bangladesh
Department of Computer Science and Engineering (CSE)

Faculty of Sciences and Engineering
Semester: (Fall, Year: 2024), B.Sc. in CSE (Day)

Lab Report NO: 03
Course Title: Data Structure Lab
Course Code: CSE 206
Section: D8

Lab Experiment Name: Implement of Bubble Sort, Insertion Sort, Selection Sort

Student Details

	Name	ID
1.	Ashab Uddin	232002274

Lab Date : **18/09/24**
Submission Date : **24/09/24**
Course Teacher's Name : **Md. Parvez Hossain**

Lab Report Status

Marks: ..... Signature: .....
Comments: ..... Date: .....

1. INTRODUCTION

The purpose of this lab report is to understand the concepts of array sorting in C programming. We will explore how to sort arrays using different sorting techniques. This report focuses on implementing and comparing three types of sorting methods to solve practical problems efficiently through programming.

2. OBJECTIVES

- To gain knowledge of various sorting algorithms and their applications.
- To develop problem-solving skills through the implementation of sorting techniques in C.
- To enhance coding proficiency by applying sorting algorithms to practical problems in C.

3. IMPLEMENTATION

Task 1: Implement Bubble Sort algorithm using Arrays

Solution:

```c
#include <stdio.h>

int main() {
    int A[100];
    int n, key;
    int swapped;

    printf("Enter the size of an array: ");
    scanf("%d", &n);

    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }

    for (int i = 0; i < n - 1; i++) {
        swapped = 0;
        for (int j = 0; j < n - i - 1; j++) {
            if (A[j] > A[j + 1]) {
                key = A[j];
                A[j] = A[j + 1];
```

```
                A[j + 1] = key;
                swapped = 1;
            }
        }
        if (!swapped) {
            break;
        }
    }

    printf("The Sorted array: \n");
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

```
PS D:\Fall 2024\DS Lab\sorting> cd "d:\Fall 2024\
Enter the size of an array: 5
Enter the elements of the array:
98 65 45 89 87
The Sorted array:
45 65 87 89 98
```

Task 2: Implement Insertion Sort algorithm using Arrays

Solution:

```c
#include <stdio.h>

void insertionSort(int A[], int n) {
    for (int i = 1; i < n; i++) {
        int key = A[i];
        int j = i - 1;


        while (j >= 0 && A[j] > key) {
            A[j + 1] = A[j];
            j = j - 1;
        }
        A[j + 1] = key;
    }
}

int main() {
    int A[100], n;


    printf("Enter the size of the array: ");
    scanf("%d", &n);


    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }


    insertionSort(A, n);
```

```
    printf("The sorted array is:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Output:

```
Enter the size of the array: 5
Enter the elements of the array:
90 58 34 09 0
The sorted array is:
0 9 34 58 90
PS D:\Fall 2024\DS Lab\sorting>
```

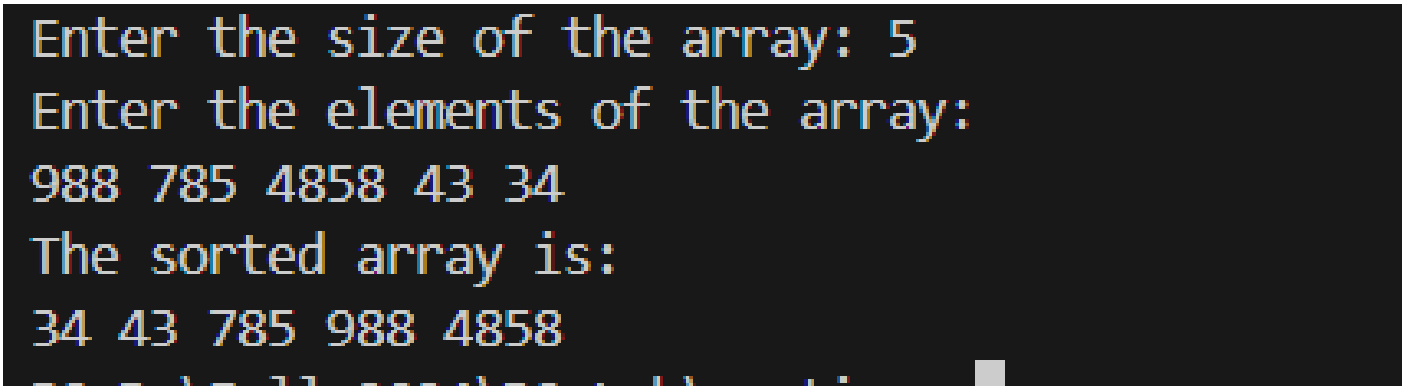
Task 3: Implement Selection Sort Algorithm Using Arrays.

Solution:

```
#include <stdio.h>

void selectionSort(int A[], int n) {
    for (int i = 0; i < n - 1; i++) {

        int minIndex = i;
        for (int j = i + 1; j < n; j++) {
            if (A[j] < A[minIndex]) {
                minIndex = j;
            }
```

```c
        }


        if (minIndex != i) {
            int temp = A[i];
            A[i] = A[minIndex];
            A[minIndex] = temp;
        }
    }
}

int main() {
    int A[100], n;


    printf("Enter the size of the array: ");
    scanf("%d", &n);


    printf("Enter the elements of the array:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &A[i]);
    }


    selectionSort(A, n);


    printf("The sorted array is:\n");
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Output:



4. DISCUSSION

In this lab report, we explored three sorting algorithms. First, **Bubble Sort**, where adjacent elements are compared and swapped if needed, repeating this process until the array is sorted. Second, **Insertion Sort**, which places each element from the unsorted portion into its correct position in the sorted portion. Lastly, **Selection Sort** repeatedly scans the array to find the smallest element and moves it to the front, continuing until the entire array is sorted. Each algorithm offers a different approach to sorting, with varying efficiency based on the data set size.