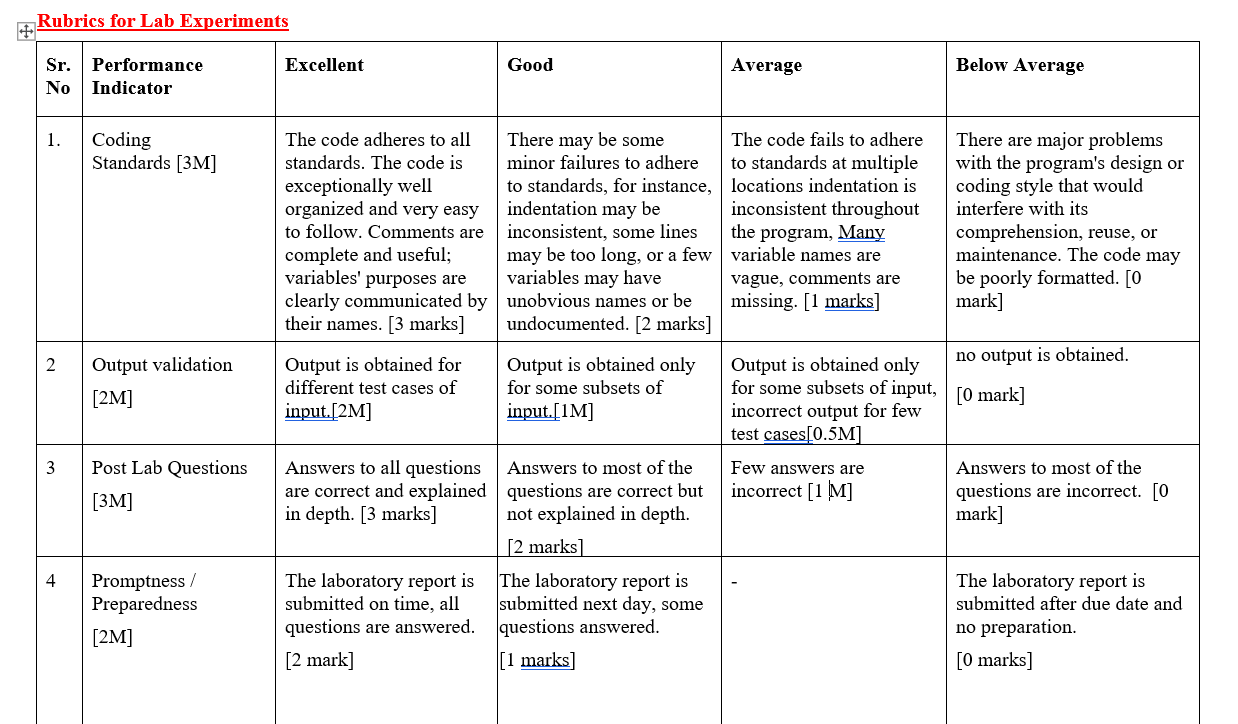
**Mark Lopes**

**S.E Comps\_A\_Batch\_C**

**9913**



**1] Bubble sort**

#include <stdio.h>

// Function to perform Bubble Sort

void bubbleSort(int arr[], int n)

{

    for (int pass = 0; pass < n - 1; pass++) // iterate through all the passes

    {

        for (int i = 0; i < n - pass - 1; i++) // iterate through all comparisons

        {

            if (arr[i] > arr[i + 1]) // if element is greater then swap

            {

                int temp = arr[i];

                arr[i] = arr[i + 1];

                arr[i + 1] = temp;

            }

        }

    }

}

// Function to print an array

void printArray(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int getArraySize(int arr[])

{

    int size = 0;

    // Iterate through the array until the end marker is found

    while (arr[size] != '\0')

    {

        size++;

    }

    return size;

}

int main()

{

    int arr[] = {34,677,87,2,45};

    int n = getArraySize(arr);

printf("Original array: ");

    printArray(arr, n);

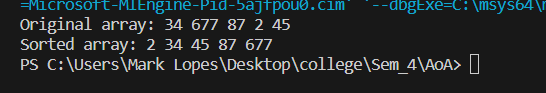
    bubbleSort(arr, n);

    printf("Sorted array: ");

    printArray(arr, n);

    return 0;

}



**2]Modified bubble sort**

#include <stdio.h>

#include <stdbool.h>

void ModifiedBubbleSort(int arr[], int n)

{

    int pass, j;

    bool exchange; // flag to check if any exchange or swaps are made between passes

    for (pass = 0; pass < n - 1; pass++)

    {

        exchange = false; // No swaps made yet in this pass

        for (j = 0; j < n - pass - 1; j++)

        {

            if (arr[j] > arr[j + 1])  // if element is greater then swap

            {

                int temp = arr[j];

                arr[j] = arr[j + 1];

                arr[j + 1] = temp;

                exchange = true; // Set the flag to indicate a swap

            }

        }

        // If no swaps were made, the array is already sorted

        if (!exchange)

        {

            break;

        }

    }

}

// Function to print an array

void printArray(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int getArraySize(int arr[])

{

    int size = 0;

    // Iterate through the array until the end marker is found

    while (arr[size] != '\0')

    {

        size++;

    }

    return size;

}

int main()

{

    int arr[] = {2,56,34,90,67};

    int n = getArraySize(arr);

printf("Original array: ");

    printArray(arr, n);

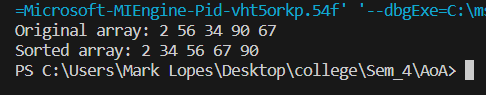
    ModifiedBubbleSort(arr, n);

    printf("Sorted array: ");

    printArray(arr, n);

    return 0;

}

****

**3] Selection sort**

#include <stdio.h>

void selectionSort(int arr[], int n)

{

    int pass, j, minIndex;

    for (pass = 0; pass < n - 1; pass++) //iterate through every pass

    {

        minIndex = pass;

        // Find the index of the minimum element in the unsorted part of the array

        for (j = pass + 1; j < n; j++)

        {

            if (arr[j] < arr[minIndex])

            {

                minIndex = j;

            }

        }

        // Swap the found minimum element with the first element and the minimum element is now in sorted part of array

        int temp = arr[pass];

        arr[pass] = arr[minIndex];

        arr[minIndex] = temp;

    }

}

// Function to print an array

void printArray(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main()

{

    int arr[] = {34,76,50,1,48};

    int n = 5;

    printf("Original array: ");

    printArray(arr, n);

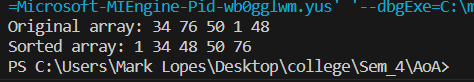
    selectionSort(arr, n);

    printf("Sorted array: ");

    printArray(arr, n);

    return 0;

}

****

**4] Insertion sort**

#include <stdio.h>

// Function to perform Insertion Sort

void insertionSort(int arr[], int n)

{

    int i, key, j;

    // Iterate through the array starting from the second element (index 1)

    for (i = 1; i < n; i++)

    {

        key = arr[i]; // Current element to be inserted in the sorted array

        j = i - 1;    // Index of the previous element

        // compare the value of key(the element to be sorted) and the element in the sorted array

        while (j >= 0 && key < arr[j])

        {

            arr[j + 1] = arr[j]; // when key(element to be inserted) is smaller than element in sorted array, replace key by element in sorted array

            j = j - 1;           // decrement the value of j to compare to the previous element(in sorted array) in next iteration

        }

        arr[j + 1] = key; // Place the key at its correct position

    }

}

int getArraySize(int arr[])

{

    int size = 0;

    // Iterate through the array until the end marker is found

    while (arr[size] != '\0')

    {

        size++;

    }

    return size;

}

// Function to print an array

void printArray(int arr[], int size)

{

    for (int i = 0; i < size; i++)

    {

        printf("%d ", arr[i]);

    }

    printf("\n");

}

int main()

{

    int arr[] = {34, 67, 99, 56, 29};

    int n = getArraySize(arr);

    printf("Original array: ");

    printArray(arr, n);

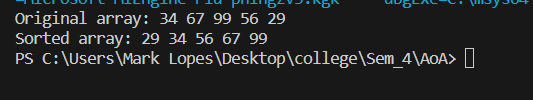
    insertionSort(arr, n);

    printf("Sorted array: ");

    printArray(arr, n);

    return 0;

}

****

