Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

I. Insertion Sort

Theory:

Insertion sort is a simple sorting algorithm that works by iteratively inserting each element of an unsorted list into its correct position in a sorted portion of the list. It is a stable sorting algorithm, meaning that elements with equal values maintain their relative order in the sorted output.

Insertion sort is like sorting playing cards in your hands. You split the cards into two groups: the sorted cards and the unsorted cards. Then, you pick a card from the unsorted group and put it in the right place in the sorted group.

Algorithm:

Programming Language:

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Code:

```
#include <iostream>
using namespace std;

int main(){
   int n;
   cout << "Enter Array Lenth : ";
   cin >> n;
   cout << endl;
   int arr[n];
   cout << "Main Array : ";
   for(int i=0; i<n ; i++){
      cin >> arr[i];
   }
   cout << "Insertion Sorted Array : ";

   for(int i=1 ; i<n ; i++){
      int current = arr[i];
      int j = i-1;
      while(arr[j]>current && j>0){
         arr[j+1] = arr[j];
         j--;
      }
      arr[j+1] = current;
      for (int i = 0; i < n; i++)
      {
         cout << arr[i] << " ";
```

```
        } cout << endl;
    }
}
```

Output:

```
Enter Array Lenth : 10

Main Array : 4
55
6
3
22
5
44
33
6
7
Insertion Sorted Array : 4 55 6 3 22 5 44 33 6 7
4 6 55 3 22 5 44 33 6 7
4 3 6 55 22 5 44 33 6 7
4 3 6 22 55 5 44 33 6 7
4 3 5 6 22 55 44 33 6 7
4 3 5 6 22 44 55 33 6 7
4 3 5 6 22 33 44 55 6 7
4 3 5 6 6 22 33 44 55 7
4 3 5 6 6 7 22 33 44 55
```

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Space complexity: _____

Justification:_____

_____

Time complexity:

Best case time complexity: _____

Justification:_____

_____

Worst case time complexity: _____

Justification:_____

_____

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

II. Bubble Sort

Theory:

Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in the wrong order. This algorithm is not suitable for large data sets as its average and worst-case time complexity is quite high.

In short it sorts elements In pair.

Algorithm:

Programming Language:

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Code:

```
#include <iostream>
using namespace std;

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

      bool swapped = false;
      for (int j = 0; j < n-i-1; j++) {
         if (arr[j] > arr[j+1]) {
            int temp = arr[j];
            arr[j] = arr[j+1];
            arr[j+1] = temp;
            swapped = true;
         }
      }
      if (!swapped)
         break;
   }
}
void printArray(int arr[], int size) {
   for (int i = 0; i < size; i++)
      cout << arr[i] << " ";
   cout << endl;
}

int main() {
```

```
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Unsorted array: ";
    printArray(arr, n);
    bubbleSort(arr, n);
    cout << "Sorted array: ";
    printArray(arr, n);
    return 0;
}
```

Output:

```
Unsorted array: 64 34 25 12 22 11 90
Sorted array: 11 12 22 25 34 64 90
```

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____
_____

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

III. Selection Sort

Theory:

Selection sort is a simple and efficient sorting algorithm that works by repeatedly selecting the smallest (or largest) element from the unsorted portion of the list and moving it to the sorted portion of the list.

The algorithm repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it with the first element of the unsorted part. This process is repeated for the remaining unsorted portion until the entire list is sorted.

Algorithm:

Programming Language:

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Code:

```
#include <iostream>
using namespace std;

void bubbleSort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        bool swapped = false;
        for (int j = 0; j < n-i-1; j++) {
            if (arr[j] > arr[j+1]) {
                int temp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = temp;
                swapped = true;
            }
        }
        if (!swapped)
            break;
    }
}
void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {64, 34, 25, 12, 22, 11, 90};
```

```
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Unsorted array: ";
    printArray(arr, n);
    bubbleSort(arr, n);
    cout << "Sorted array: ";
    printArray(arr, n);
    return 0;
}
```

Output:

```
Unsorted array: 54 35 2 22 51
Sorted array: 2 22 35 51 54
```

Space complexity: _____

Justification:_____
_____

Time complexity:

Best case time complexity: _____

Justification:_____
_____

Worst case time complexity: _____

Justification:_____
_____

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

IV. Counting Sort

Theory:

Counting sort is a sorting technique that is based on the keys between specific ranges. In coding or technical interviews for software engineers, sorting algorithms are widely asked. So, it is important to discuss the topic.

This sorting technique doesn't perform sorting by comparing elements. It performs sorting by counting objects having distinct key values like hashing. After that, it performs some arithmetic operations to calculate each object's index position in the output sequence. Counting sort is not used as a general-purpose sorting algorithm.

Counting sort is effective when range is not greater than number of objects to be sorted. It can be used to sort the negative input values.

Algorithm:

Programming Language:

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Code:

```
#include <iostream>
#include <vector>
using namespace std;
void countingSort(int arr[], int n) {

   int max = arr[0];
   for (int i = 1; i < n; i++) {
      if (arr[i] > max)
         max = arr[i];
   }
   vector<int> count(max + 1, 0);

   for (int i = 0; i < n; i++) {
      count[arr[i]]++;
   }
   for (int i = 1; i <= max; i++) {
      count[i] += count[i - 1];
   }

   int output[n];
   for (int i = n - 1; i >= 0; i--) {
      output[count[arr[i]] - 1] = arr[i];
      count[arr[i]]--;
   }
   for (int i = 0; i < n; i++) {
      arr[i] = output[i];
```

```
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        cout << arr[i] << " ";
    cout << endl;
}

int main() {
    int arr[] = {4, 2, 2, 8, 3, 3, 1};
    int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Unsorted array: ";
    printArray(arr, n);
    countingSort(arr, n);
    cout << "Sorted array: ";
    printArray(arr, n);
    return 0;
}
```

Output:

```
Unsorted array: 4 2 2 8 3 3 1
Sorted array: 1 2 2 3 3 4 8
```

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the sorting algorithms	
Experiment No: 1	Date:	Enrolment No: 92200133002

Space complexity: _____

Justification:_____

_____

Time complexity:

Best case time complexity: _____

Justification:_____

_____

Worst case time complexity: _____

Justification:_____

_____