DATA STRUCTURES

DAY-11

1.Insettion Sorting

Program:

```
#include <stdio.h>
void insertionSort(int arr[], int n) {
   for (int i = 1; i < n; i++) {
      int key = arr[i];
      int j = i - 1;
       while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
void printArray(int arr[], int size) {
   for (int i = 0; i < size; i++) {
      printf("%d ", arr[i]);
   }
   printf("\n");
}
int main() {
   int arr[] = {12, 11, 13, 5, 6};
   int n = sizeof(arr) / sizeof(arr[0]);
```

```
    printf("Original array: \n");
    printArray(arr, n);

    insertionSort(arr, n);
    printf("Sorted array: \n");
    printArray(arr, n);
    return 0;
}
```

Output:

Original array:

12 11 13 5 6

Sorted array:

5 6 11 12 13

2.Merge sort

Program:

```
#include <stdio.h>
void merge(int arr[], int left, int mid, int right) {
    int n1 = mid - left + 1;
    int n2 = right - mid;
    int L[n1], R[n2];
    for (int i = 0; i < n1; i++)
        L[i] = arr[left + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[mid + 1 + j];
     int i = 0;
```

```
    int j = 0;
    int k = left;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
        k++;
    }
     while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }
    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}
void mergeSort(int arr[], int left, int right) {
    if (left < right) {
            int mid = left + (right - left) / 2;
```

```
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
         merge(arr, left, mid, right);
    }
}

void printArray(int arr[], int size) {
    for (int i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
 int main() {
    int arr[] = {12, 11, 13, 5, 6, 7};
    int arr_size = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is \n");
    printArray(arr, arr_size);
    mergeSort(arr, 0, arr_size - 1);
    printf("\nSorted array is \n");
    printArray(arr, arr_size);
    return 0;
}
```

Output:

Given array is

12 11 13 5 6 7

Sorted array is

5 6 7 11 12 13

3.Radix Sort

Program:

```
#include <stdio.h>
 int getMax(int arr[], int n) {
   int max = arr[0];
   for (int i = 1; i < n; i++)
      if (arr[i] > max)
         max = arr[i];
   return max;
}
void countSort(int arr[], int n, int exp) {
   int output[n]; // output array
   int i, count[10] = {0};
    for (i = 0; i < n; i++)
      count[(arr[i] / exp) % 10]++;
   for (i = 1; i < 10; i++)
      count[i] += count[i - 1];
    for (i = n - 1; i >= 0; i--) {
      output[count[(arr[i] / exp) % 10] - 1] = arr[i];
      count[(arr[i] / exp) % 10]--;
   }
    for (i = 0; i < n; i++)
      arr[i] = output[i];
}
void radixSort(int arr[], int n) {
      int m = getMax(arr, n);
```

```
    for (int exp = 1; m / exp > 0; exp *= 10)
        countSort(arr, n, exp);
}
 void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}

int main() {
    int arr[] = {170, 45, 75, 90, 802, 24, 2, 66};
    int n = sizeof(arr) / sizeof(arr[0]);
    printf("Given array is \n");
    printArray(arr, n);
    radixSort(arr, n);
    printf("Sorted array is \n");
    printArray(arr, n);
    return 0;
}
```

Output:

Given array is

170 45 75 90 802 24 2 66

Sorted array is

2 24 45 66 75 90 170 802