DATASTRUCTURES

7/08/24

1.INSERTION SORT:

PROGRAMME

```
#include <stdio.h>
 void insertionSort(int arr[], int n)
{
   int i, key, j;
   for (i = 1; i < n; i++)
{
       key = arr[i];
       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) {
   int i;
   for (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:

```
#include <stdio.h>
#include <stdlib.h>
void merge(int arr[], int l, int m, int r) {
    int n1 = m - l + 1; // Size of the left subarray
    int n2 = r - m;     // Size of the right subarray
     int *L = (int *)malloc(n1 * sizeof(int));
    int *R = (int *)malloc(n2 * sizeof(int));
     for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
```

```
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    int i = 0;
    int j = 0;
    int k = l;

    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k++] = L[i++];
        } else {
            arr[k++] = R[j++];
        }
    }
    while (i < n1) {
        arr[k++] = L[i++];
    }
     while (j < n2) {
        arr[k++] = R[j++];
    }
     free(L);
    free(R);
}
void mergeSort(int arr[], int l, int r) {
    if (l < r) {
        int m = l + (r - l) / 2;
```

```
        mergeSort(arr, l, m);
        mergeSort(arr, m + 1, r);
        merge(arr, l, m, r);
    }
}
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("Original array: \n");
    printArray(arr, arr_size);

    mergeSort(arr, 0, arr_size - 1);

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

OUTPUT:

Original array:

12 11 13 5 6 7

Sorted array:

5 6 7 11 12 13