SORTING-QUICK SORT

AIM: THE AIM OF THE PROGRAM IS TO IMPLEMENT QUICK SORT-SORTING ALGORITHM

ALGORITHM: Step-by-Step Algorithm
Step 1: Select a Pivot
Any element at random.
The first or last element.
Middle element.
Step 2: Rearrange the Array
The pivot element is compared to all of the items starting with the first index. If the element is greater than the pivot element, a second pointer is appended.
When compared to other elements, if a smaller element than the pivot element is found, the smaller element is swapped with the larger element identified before.
Step 3: Divide the Subarrays

sort the segment of the array to the left of the pivot, and then sort the segment of the array to the right of the pivot.
Step 4: Final Sorted Array

After sorting both subarrays and placing the pivot in its correct position, we combine the elements to form the sorted array

PROGRAM:
```
#include <stdio.h>
int partition(int arr[], int low, int high)
{
   int pivot = arr[high];
   int i = (low - 1);
   for (int j = low; j <= high - 1; j++)
   {
      if (arr[j] < pivot)
      {
         i++;
         int temp = arr[i];
         arr[i] = arr[j];
         arr[j] = temp;
      }
   }
   int temp = arr[i + 1];
   arr[i + 1] = arr[high];
   arr[high] = temp;
   return (i + 1);
```

```
}
void quickSort(int arr[], int low, int high)
{
    if (low < high)
    {
        int pivot = partition(arr, low, high);
        quickSort(arr, low, pivot - 1);
        quickSort(arr, pivot + 1, high);
    }
}
int main()
{
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("\nInitially array elemnts are:\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    quickSort(arr, 0, n - 1);
    printf("\nSorted array using Quick Sort:\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:

```
Original array - 2 11 9 4 13 5
Array after sorting - 2 4 5 9 11 13
```

RESULT:

THE PROGRAM IMPLEMENTS THE QUICK SORT-SORTING ALGORITHM SUCCESSFULLY.

MERGE SORT

AIM: THE AIM OF THE PROGRAM IS TO IMPLEMENT MERGE SORT-SORTING ALGORITHM

ALGORITHM: Step-by-Step Algorithm

If the list has 0 or 1 elements, it is already sorted. Return the list.
Find the middle index of the list.
Recursively apply merge sort to the left half of the list.
Recursively apply merge sort to the right half of the list.
Create an empty result list to store the merged elements.
Initialize two pointers for the left and right halves, starting at the first element of each half.
Compare the elements at the pointers in the left and right halves.
Add the smaller element to the result list and move the corresponding pointer.
Continue comparing and adding elements until all elements from one half are added to the result list.
Add any remaining elements from the other half to the result list and return it.
PROGRAM:

```
#include <stdio.h>
void merge(int arr[], int l, int m, int r)
{
    int i, j, k;
    int n1 = m - l + 1;
    int n2 = r - m;
    int L[n1], R[n2];
    for (i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];
    i = 0;
    j = 0;
    k = l;
    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 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);
    }
}
int main()
{
    int n;
    printf("Enter the number of elements: ");
    scanf("%d", &n);
    int arr[n];
    for (int i = 0; i < n; i++)
    {
        printf("Enter element %d: ", i + 1);
        scanf("%d", &arr[i]);
    }
    printf("\nInitially array elements are:\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
```

```
    mergeSort(arr, 0, n - 1);
    printf("\nSorted array using Merge Sort:\n");
    for (int i = 0; i < n; i++)
    {
        printf("%d ", arr[i]);
    }
    return 0;
}
```

OUTPUT:

Original array - 2 11 9 4 13 5
Array after sorting - 2 4 5 9 11 13

RESULT:

THE PROGRAM IMPLEMENTS THE MERGE SORT-SORTING ALGORITHM SUCCESSFULLY.