**AIM:**

Write a C program to take n numbers and sort the numbers in ascending order .Try to implement the same using following sorting techniques.

1. Quick Sort

2. Merge Sort

**QUICK SORT:**

**ALGORITHM:**

QuickSort(A[], LEFT, RIGHT)

A[] : Integer array

LEFT, RIGHT : Integer

Step 1 : Start.

Step 2 : If LEFT < RIGHT then Go to Step 3 else Go to Step 23.

Step 3 : Set PIVOT = LEFT.

Step 4 : Set I = LEFT + 1.

Step 5 : Set J = RIGHT.

Step 6 : Repeat While I < J.

Step 7 : Repeat While A[I] < A[PIVOT].

Step 8 : Increment I by 1.

Step 9 : [End of Step 7 While loop].

Step 10 : Repeat While A[J] > A[PIVOT].

Step 11 : Decrement J by 1.

Step 12 : [End of Step 10 While loop].

Step 13 : If I < J then Go to Step 14 else Go to Step 17.

Step 14 : Set TEMP = A[I].

Step 15: Set A[I] = A[J].

Step 16 : Set A[J] = TEMP.

Step 17 : [End of Step 6 While loop].

Step 18 : Set TEMP = A[PIVOT].

Step 19 : Set A[PIVOT] = A[J].

Step 20 : Set A[J] = TEMP.

Step 21 : QUICKSORT(LEFT, J – 1),

Step 22 : QUICKSORT(J + 1, RIGHT).

Step 23 : Stop.

**PROGRAM:**

#include <stdio.h>

void QuickSort(int a[], int left, int right);

int main()

{

int i, n, a[10];

printf("Enter the limit : ");

scanf("%d", &n);

printf("Enter the elements : ");

for (i = 0; i < n; i++)

scanf("%d", &a[i]);

QuickSort(a, 0, n - 1);

printf("The sorted elements are : ");

for (i = 0; i < n; i++)

printf("%d\t", a[i]);

return 0;

}

void QuickSort(int a[], int left, int right)

{

int i, j, temp, pivot;

if (left < right) {

pivot = left;

i = left + 1;

j = right;

while (i < j)

{

while (a[i] < a[pivot])

i++;

while (a[j] > a[pivot])

j--;

if (i < j) {

temp = a[i];

a[i] = a[j];

a[j] = temp;

}

}

temp = a[pivot];

a[pivot] = a[j];

a[j] = temp;

QuickSort(a, left, j - 1);

QuickSort(a, j + 1, right);

}

}

**OUTPUT:**

Enter the limit : 5

Enter the numbers : 30 40 50 20 10

The sorted elements are : 10 20 30 40 50

**MERGE SORT:**

**ALGORITHM:**

MERGE (ARR [], LEFT, CENTER, RIGHT)

A [] : Integer array

LEFT : Integer

CENTER : Integer

RIGHT : Integer

Step 1 : Start.

Step 2 : Set N1 = CENTER - LEFT + 1.

Step 3 : Set N2 = RIGHT - CENTER.

Step 4 : Repeat For I = 0 to N1 - 1.

Step 5 : Set A[I] = ARR[LEFT + I].

Step 6 : Increment I by 1.

Step 7 : [End of Step 4 For loop].

Step 8 : Repeat For J = 0 to N2 - 1.

Step 9 : Set B[J] = ARR[CENTER + 1 + J].

Step 10 : Increment J by 1.

Step 11 : [End of Step 8 For loop].

Step 12 : Repeat While APTR < N1 AND BPTR < N2.

Step 13 : If A[APTR]<= B[BPTR] then Go to Step 14 else Go to Step 18.

Step 14 : Set ARR[CPTR] = A[APTR].

Step 15 : Increment APTR by 1 and Go to Step 19.

Step 16 : Set ARR[CPTR] = B[BPTR].

Step 17 : Increment BPTR by 1.

Step 18 : Increment CPTR by 1.

Step 19 : [End of Step 12 While loop].

Step 20 : Repeat While APTR < N1.

Step 21 : Set ARR[CPTR] = A[APTR].

Step 22 : Increment APTR by 1.

Step 23 : Increment CPTR by 1.

Step 24 : [End of Step 20 While loop].

Step 25 : Repeat While BPTR < N2.

Step 26 : Set ARR[CPTR] = B[BPTR].

Step 27 : Increment BPTR by 1.

Step 28 : Increment CPTR by 1.

Step 29 : [End of Step 25 While loop].

Step 30 : Stop.

**PROGRAM:**

#include <stdio.h>

#include <stdlib.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); }

}

void printArray(int A[], int size)

{

int i;

for (i = 0; i < size; i++)

printf("%d ", A[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:**

Enter the limit : 5

Enter the numbers : 30 40 50 20 10

The sorted elements are : 10 20 30 40 50

**RESULT:**

Hence, the program of sorting the numbers using quick sort and merge sort is implemented.