**DAY 12 – QUICK SORT & MERGE SORT**

15. Write a menu driven C program to implement (i) Quick sort.

(ii) Merge sort

**PROGRAM**

#include<stdio.h>

#define MAX 10

int i, j, n, a[MAX];

void read()

{

    printf("\nEnter no. of elements in the array: ");

    scanf("%d", &n);

    printf("\nEnter the elements: ");

    for(i = 0; i < n; i++)

        scanf("%d", &a[i]);

}

void display()

{

 for(i = 0; i < n; i++)

    printf("%d\t", a[i]);

}

int partition(int *beg*, int *end*)

{

    int left, right, temp, loc, flag;

    loc = left = *beg*;

    right = *end*;

    flag = 0;

    while(flag != 1)

    {

        while((a[loc] <= a[right]) && (loc != right))

        right--;

        if(loc == right)

        flag = 1;

        else if(a[loc] > a[right])

        {

            temp = a[loc];

            a[loc] = a[right];

            a[right] = temp;

            loc = right;

        }

        if(flag != 1)

        {

            while((a[loc] >= a[left]) && (loc != left))

                left++;

            if(loc == left)

                flag = 1;

            else if(a[loc] < a[left])

            {

                temp = a[loc];

                a[loc] = a[left];

                a[left] = temp;

                loc = left;

            }

        }

    }

    return loc;

}

void quick\_sort(int *beg*, int *end*)

{

    int loc;

    if(*beg*<*end*)

    {

        loc = partition(*beg*, *end*);

        quick\_sort(*beg*, loc - 1);

        quick\_sort(loc + 1, *end*);

    }

}

void merge(int *beg*, int *mid*, int *end*)

{

    int index = *beg*, temp[MAX],k;

    i = *beg*, j = *mid* + 1;

    while((i <= *mid*) && (j <= *end*))

    {

        if(a[i] < a[j])

        {

            temp[index] = a[i];

            i++;

        }

        else

        {

            temp[index] = a[j];

            j++;

        }

        index++;

    }

    if(i > *mid*)

    {

        while(j <= *end*)

        {

            temp[index] = a[j];

            j++;

            index++;

        }

    }

    else

    {

        while(i <= *mid*)

        {

            temp[index] = a[i];

            i++;

            index++;

        }

    }

    for(k = *beg*; k < index; k++)

        a[k] = temp[k];

}

void merge\_sort(int *beg*, int *end*)

{

    int mid;

    if(*beg* < *end*)

    {

        mid = (*beg* + *end*) / 2;

        merge\_sort(*beg*, mid);

        merge\_sort(mid + 1, *end*);

        merge(*beg*, mid, *end*);

    }

}

int main()

{

    int ch;

    do

    {

        printf("\n\t\tMENU\n");

        printf("1. Entry\n2. Display\n3. Quick Sort\n4. Merge Sort\n5. Exit\n");

        printf("Enter choice: ");

        scanf("%d", &ch);

        switch(ch)

        {

        case 1: read();

                break;

        case 2: printf("Array:\t");

                display();

                break;

        case 3: quick\_sort(0, n - 1);

                printf("Sorted array after quick sort: ");

                display();

                break;

        case 4: merge\_sort(0, n - 1);

                printf("Sorted array after merge sort: ");

                display();

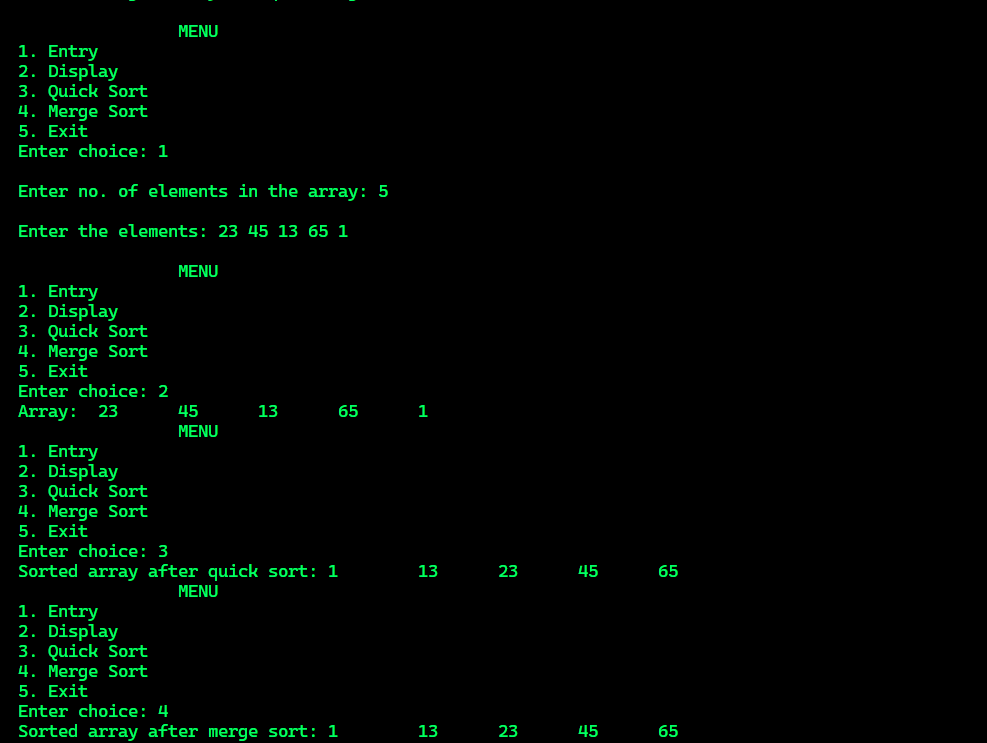
                break;

        }

    } while (ch >= 1 && ch <= 4);

}

**OUTPUT**

****