```
#include<stdio.h>

#define n 7

int a[50] ,b[50];
int array[50] ,len;

void printarray(int arr[] ,int len){
        for(int i=0 ;i<len ;i++){
                printf("%d\t",arr[i]);
        }
}

void swap(int i,int j,int arr[]){
        int tmp;
        tmp=arr[i];
        arr[i]=arr[j];
        arr[j]=tmp;
}

//merge sort
void mergesort(int low, int mid, int high, int arr[] ){

        int k=low,i=low,j=mid+1;
        while((i<=mid) && (j<=high)){
                if(arr[i] <= arr[j]){
                        b[k]=arr[i];
                        i++;
                }
                else{
                        b[k]=arr[j];
                        j++;
                }
                k++;
        }
        while(i<=mid){
                b[k]=arr[i];
                k++;
                i++;
        }
        while(j<=high){
                b[k]=arr[j];
                k++;
                j++;
        }
        for(i=low; i<=high; i++)
                arr[i]=b[i];

        printf("\t");
        printarray(arr,len);
```

```
        printf("\n");
}
void mergesplit(int arr[], int low, int hi){
        if(low < hi){
                int mid=(low+hi)/2;
                mergesplit(arr ,low ,mid);
                mergesplit(arr ,mid+1 ,hi);
                mergesort(low ,mid ,hi ,arr);
        }
}


//quicksort
void quicksort(int arr[], int first, int last){
    if(first<last){
        int i,j,pivot;
        i=first;
        j=last;
        pivot=arr[first];
        while(i<j){
            while(arr[i]<=pivot && i<last)
                i++;
            while(arr[j]>=pivot && j>first)
                j--;
            if(i<j)
                swap(i,j,arr);
        }
        swap(j,first,arr);

                printf("\t");
                printarray(arr,len);
                printf("\n");

        quicksort(arr,first,j-1);
        quicksort(arr,j+1,last);
    }
}

int main(){
        len=8;
        int array[]={100,5,3,8,2,7,9,1};
//      printf("enter the array size : ");
//      scanf("%d",&len);

//      printf("enter the array elements : ");
//      for(int i=0;i<len;i++)
//          scanf("%d",&array[i]);

    int choice, pos;
```

```
    printf("\n1...merge sort\n");
    printf("2...quick sort\n");
    printf("3...quit\n");
    int quit=1;
    while(quit!=0){
        printf("\nOption : ");
        scanf("%d",&choice);
        for(int i=0; i<len; i++)
        {
                a[i]=array[i];
        }
        switch(choice){
            case 1: printf("before merge sort : ");
                printarray(a,len);
                printf("\n");
                mergesplit(a ,0 ,len-1);
                break;
            case 2: printf("before quick sort : ");
                printarray(a,len);
                printf("\n");
                quicksort(a ,0 ,len-1);
                break;
            case 3: printf("***program terminated*** ");
                quit=0;
                break;
            default:
                            printf("\n1...merge sort\n");
                            printf("2...quick sort\n");
                            printf("3...quit\n");
        }
    }

        return 0;
}
```

```
Option : 4
before merge sort : 100 5   3   8   2   7   9   1
    5   100 3   8   2   7   9   1
    5   100 3   8   2   7   9   1
    3   5   8   100 2   7   9   1
    3   5   8   100 2   7   9   1
    3   5   8   100 2   7   1   9
    3   5   8   100 1   2   7   9
    1   2   3   5   7   8   9   100

Option : 5
before quick sort : 100 5   3   8   2   7   9   1
    1   5   3   8   2   7   9   100
    1   5   3   8   2   7   9   100
    1   2   3   5   8   7   9   100
    1   2   3   5   8   7   9   100
    1   2   3   5   7   8   9   100
```

Sorting Algo

Algorithm

I. void printarray (int arr[], int len)

1. START
2. for (int i = 0 to len)
    1. print (arr[i] + "\t")
3. End for
4. STOP

I void mergesort (int low, int mid, int high, int arr[])

1. START
2. int k=low, i=low, j=mid+1
3. while ((i<=mid) && (j<=high))
    1. if (arr[i] <= arr[j])
        1. b[k]=arr[i];
        2. i++
    2. Else
        1. b[k] = arr[j];
        2. j++
    3. End If
    4. k++
4. End While
5. While (i<=mid)
    1. b[k] = arr[i]
    2. k++
    3. i++
6. End While
7. While (j<=high)
    1. b[k] = arr[j]
    2. k++
    3. j++
8. End While

9. for(int i = low to high)
    1. arr[i] = b[i]
10. printarray (arr, len)
11. STOP

III void merge (int arr[], int low, int hi)

1. START
2. if (low < hi)
    1. int mid = (low + hi) / 2;
    2. merge (arr, low, mid)
    2. merge (arr, mid+1, high)
    3. mergesort(low, mid, hi, arr)
3. End If
4. STOP

IV ~~void merge~~ void quicksort (int arr[], int first, int last)

1. START
2. if (first < last)
    1. int i, j, pivot
    2. i = first
    3. j = last
    4. pivot = arr[first]
    5. while (i < j)

1. while (arr[i] <= pivot && i < last)
    1. i++
2. while (arr[j] >= pivot && j > first)
    2. j--
3. if (i < j)
    1. swap (i, j, arr);
4. End If
6. swap (j, first, arr)
7. printarray (arr, len)
8. quicksort (arr, first, j-1)
9. quicksort (arr, j+1, last)

3. Else If
4. STOP

V void main()

1. START
2. create an array
3. call merge() & quicksort()
4. STOP