PRACTICE 2

PRACTICE:2.1

AIM: Write a C program on Quick sort

PROGRAM:

```
#include <stdio.h>
void quicksort(int array[], int first, int last)
{
 int i, j, pivot, temp;
   if (first < last)
        {
      pivot = first; // Choosing first element as pivot
      i = first;
      j = last;
   while (i < j)
        {
         while (array[i] <= array[pivot] && i < last)
            i++;
         while (array[j] > array[pivot])
            j--;
         if (i < j) { // Swap elements at i and j
            temp = array[i];
            array[i] = array[j];
            array[j] = temp;
         }
      }
      // Swap pivot with array[j] to place it in the correct position
```

```
        temp = array[pivot];
        array[pivot] = array[j];
        array[j] = temp;
        // Recursively sort the sub-arrays
        quicksort(array, first, j - 1);
        quicksort(array, j + 1, last);
    }
}
int main() {
    int i, n, array[25];
    printf("How many elements are you going to enter? ");
    scanf("%d", &n);
    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
    scanf("%d", &array[i]);
    quicksort(array, 0, n - 1);
         printf("The sorted elements are:\n");
    for (i = 0; i < n; i++)
    printf("%d ", array[i]);
    return 0;
}
```

OUTPUT:

How many elements are you going to enter? 5

Enter 5 elements: 3 1 2 5 4

The sorted elements are:

1 2 3 4 5

PRACTICE:2.1

AIM: Write a C program on Merge sort

PROGRAM:

```
#include <stdio.h>
void merge(int arr[], int low, int mid, int high)
{
   int b[10];
   int i = low, j = mid + 1, k = 0;
   while (i <= mid && j <= high) {
      if (arr[i] < arr[j])
                  {
         b[k] = arr[i];
         i++;
       }
                   else
                  {
         b[k] = arr[j];
         j++;
       }
      k++;
   }

   // Copy remaining elements from left subarray
   while (i <= mid)
        {
      b[k] = arr[i];
```

```
        i++;
        k++;
    }
        // Copy remaining elements from right subarray
    while (j <= high)
        {
        b[k] = arr[j];
        j++;
        k++;
    }
    // Copy merged elements back to original array
    for (i = low, k = 0; i <= high; i++, k++)
        arr[i] = b[k];
}
// Function to perform merge sort
void mergeSort(int arr[], int left, int right)
 {
    if (left < right)
        {
        int mid = (left + right) / 2; // Find the middle point
        // Recursively sort first and second halves
        mergeSort(arr, left, mid);
        mergeSort(arr, mid + 1, right);
        // Merge the sorted halves
        merge(arr, left, mid, right);
    }
```

```
}
// Main function
int main()
{
   int a[25],i,n;
   printf("enter n value");
   scanf("%d",&n);
   printf("enter %d elements\n",n);
   for(i=0;i<n;i++)
   scanf("%d",&a[i]);
   mergeSort(a, 0, n - 1);
   printf("Sorted array:\n");
        for(i=0;i<n;i++)
        printf("%d\t",a[i]);
   return 0;
}
```

OUTPUT:

```
enter n value5
enter 5 elements
4 3 2 5 1
Sorted array:
1       2       3       4       5
```

PRACTICE:2.3

AIM: Write a C program on Radix sort

PROGRAM:

```
#include <stdio.h>
// Function to implement radix sort
void radixSort(int a[], int n)
{
int big, nod = 0,steps,count[10], i, j, k,bucket[10][n],loc,div = 1;
  big = a[0], i;
   for(i = 1; i < n; i++)
   {
      if(a[i] > big)
         big = a[i];
          }
  // Count the number of digits in the largest number
  while (big > 0)
  {
    nod++;
    big = big / 10;
  }

  for (steps = 1; steps <= nod; steps++) {
    // Initialize count array
    for (j = 0; j < 10; j++) {
      count[j] = 0;
    }
```

```
    // Distribute elements into buckets
    for (i = 0; i < n; i++) {
      loc = (a[i] / div) % 10;
      bucket[loc][count[loc]++] = a[i];
    }
    // Collect elements back into the array
    k = 0;
    for (j = 0; j < 10; j++) {  // Looping through digits 0-9
      for (i = 0; i < count[j]; i++) {
        a[k] = bucket[j][i];
        k++;
      }
    }
    // Move to the next digit
    div = div * 10;
  }
}
int main() {
  int a[100],n,i;
  printf("\nEnter No.of elements ");
  scanf("%d",&n);
  printf("enter %d elements",n);
  for (i=0;i<n;i++) {
    scanf("%d",&a[i]);
  }
```

```
  radixSort(a, n);
  printf("After applying Radix sort, array elements are:\n");
  for (i=0;i<n;i++)
    printf("%d ",a[i]);
  return 0;
  }
```

OUTPUT:

Enter No.of elements 5

enter 5 elements3 4 1 2 5

After applying Radix sort, array elements are:

1 2 3 4 5