NAME: JAYMIN VALAKI D
ROLL NO: MA075

Practical 10

Sorting Algorithm

1. Bubble Sort

CODE:

```
#include<stdio.h>
#include<stdlib.h>
//Bubble Sort
int main()
{
   int n;
   printf("Enter Number Of Elements Of Array:");
   scanf("%d",&n);
   int a[n];
   printf("Enter Array :\n");
   for(int i=0;i<n;i++)
   {
      scanf("%d",&a[i]);
   }
   printf("\n---Before Sorting---\n");
   for(int i=0;i<n;i++)
   {
      printf("%d\n",a[i]);
   }
   printf("\n---After Bubble Sort---\n");
   for(int i=0;i<n;i++)
```

```
    {
        for(int j=0;j<n-i-1;j++)
        {
            if(a[j]>a[j+1])
            {
                int temp;
                temp=a[j+1];
                a[j+1]=a[j];
                a[j]=temp;
            }
        }
    }
    for(int i=0;i<n;i++)
    {
        printf("%d\n",a[i]);
    }
    return 0;
}
```

OUTPUT:

```
Enter Number Of Elements Of Array:10
Enter Array :
7
3
89
2052
-5
0
12
0
0
-6

---Before Sorting---
7
3
89
2052
-5
0
12
0
0
-6

---After Bubble Sort---
-6
-5
0
0
0
3
7
12
89
2052
```

2. Insertion Sort

CODE:

```
#include <math.h>
#include <stdio.h>
//Insertion Sort

void insertionSort(int arr[], int no)
{
    int i, key, j;
    for (i = 1; i < no; i++) {
            key = arr[i];
            j = i - 1;
      while (j >= 0 && arr[j] > key) {
                    arr[j + 1] = arr[j];
                    j = j - 1;
            }
            arr[j + 1] = key;
    }
}
```

```
void printArray(int arr[], int no)
{
    int i;
    printf("\n---After Sorting---\n");
    for (i = 0; i < no; i++)
            printf("%d\n", arr[i]);
    printf("\n");
}

int main()
{
   int no;
   printf("Enter Number :");
   scanf("%d",&no);
   int arr[no];
   printf("Enter Array:\n");
   for(int i=0;i<no;i++)
   {
      scanf("%d",&arr[i]);
   }

    insertionSort(arr, no);
    printArray(arr, no);

    return 0;
}
```

OUTPUT:

3. Merge Sort

CODE:

```
#include <stdio.h>
void merge(int arr[], int p, int q, int r)
{

   int n1 = q - p + 1;
   int n2 = r - q;

   int L[n1], M[n2];

   for (int i = 0; i < n1; i++)
      L[i] = arr[p + i];
   for (int j = 0; j < n2; j++)
      M[j] = arr[q + 1 + j];

   int i, j, k;
```

```
i = 0;
j = 0;
k = p;

while (i < n1 && j < n2)
{
   if (L[i] <= M[j])
   {
      arr[k] = L[i];
      i++;
   }
   else
   {
      arr[k] = M[j];
      j++;
   }
   k++;
}

while (i < n1)
{
   arr[k] = L[i];
   i++;
   k++;
}
```

```
    while (j < n2)
    {
        arr[k] = M[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 arr[], int size)
{
    for (int i = 0; i < size; i++)
        printf ("%d\n", arr[i]);
    printf("\n");
}
int main ()
```

```
{
   int n;
   printf("Enter Number :");
   scanf("%d", &n);
   int arr[n];
   printf("Enter Array Elements :\n");
   for (int i = 0; i < n; i++)
      scanf("%d", &arr[i]);
   int size = sizeof(arr) / sizeof(arr[0]);
   mergeSort(arr, 0, size - 1);
   printf("Sorted array: \n");
   printArray(arr, size);
```

}**OUTPUT:**

4. Quick Sort

CODE:

```
#include <stdio.h>
```

```
void quicksort (int number [25], int first, int last)
{
   int i, j, pivot, temp;
   if (first < last)
   {
      pivot = first;
      i = first;
      j = last;
      while (i < j)
      {
         while (number[i] <= number[pivot] && i < last)
            i++;
         while (number[j] > number[pivot])
            j--;
         if (i < j)    {
            temp = number[i];
            number[i] = number[j];
            number[j] = temp;
      }}
      temp = number[pivot];
      number[pivot] = number[j];
      number[j] = temp;
      quicksort(number, first, j - 1);
      quicksort(number, j + 1, last);
   }
}
```

```
int main ()
{
   int i, count, number [100];
   printf ("Enter Number of Elements:");
   scanf ("%d", &count);
   printf ("Enter %d elements: ", count);
   for (i = 0; i < count; i++)
      scanf ("%d", &number[i]);
   quicksort (number, 0, count - 1);
   printf ("Order of Sorted elements:\n");
   for (i = 0; i < count; i++)
      printf (" %d\n", number[i]);
   return 0;
}
```

OUTPUT: