WEEK-4

Date:16-07-2025

List of programs:

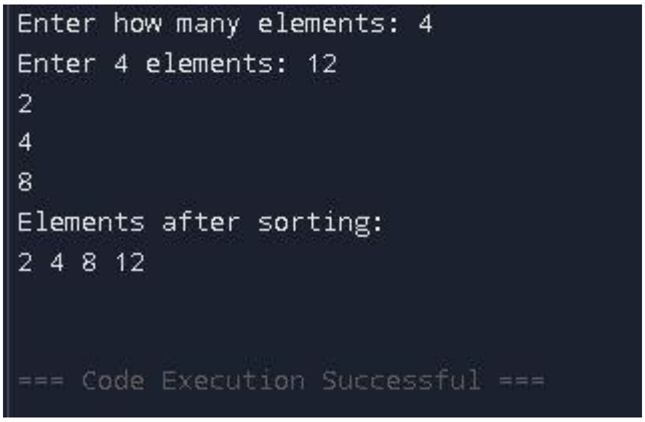
1. Write a C program to sort a given list of integers using Quick Sort Technique.
2. Write a C program to sort a given list of integers using Merge Sort Technique

1. **Aim:** To write a C program to sort a given list of integers using Quick Sort Technique.

Program:

```
#include <stdio.h>
void quicksort(int A[], int low, int high);
int partition(int A[], int low, int high);
void quicksort(int A[], int low, int high)
{
    if (low < high)
    {
        int m = partition(A, low, high);
        quicksort(A, low, m - 1);
        quicksort(A, m + 1, high);
    }
}

int partition(int A[], int low, int high)
{
    int pivot = A[low];
    int i = low + 1;
    int j = high;
    int temp;
    while (i < j)
    {
        while ( A[i] <= pivot)
            i++;
        while (A[j] > pivot)
            j--;

        if (i < j)
        {
            temp = A[i];
            A[i] = A[j];
            A[j] = temp;
        }
    }
    A[low] = A[j];
    A[j] = pivot;
    pivot=temp;
    return j;
}

int main()
{
    int i, n, A[20];
    printf("Enter how many elements: ");
    scanf("%d", &n);

    printf("Enter %d elements: ", n);
    for (i = 0; i < n; i++)
    {
        scanf("%d", &A[i]);
    }

    quicksort(A, 0, n - 1);

    printf("Elements after sorting:\n");
    for (i = 0; i < n; i++)
    {
        printf("%d ", A[i]);
    }
    printf("\n");

    return 0;
}
```

Output:



```
Enter how many elements: 4
Enter 4 elements: 12
2
4
8
Elements after sorting:
2 4 8 12


=== Code Execution Successful ===
```

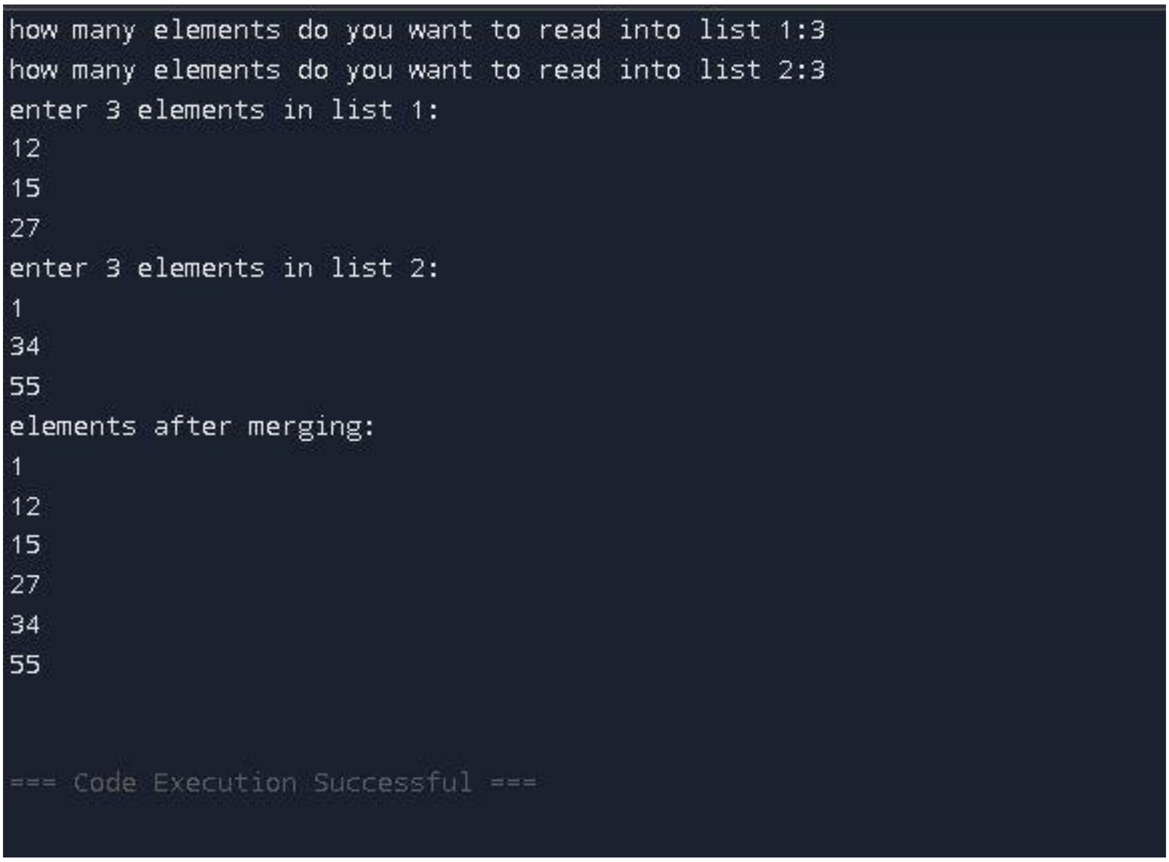
2. **Aim:** Write a C program to sort a given list of integers using Merge Sort Technique.

Program:

```
#include<stdio.h>
int main()
{
  int a[100],b[100],c[100],m,n,i,j,k;
  printf("how many elements do you want to read into list 1:");
  scanf("%d",&m);
  printf("how many elements do you want to read into list 2:");
  scanf("%d",&n);
  printf("enter %d elements in list 1:\n",m);
  for(i=0;i<m;i++)
  {
    scanf("%d",&a[i]);
  }
  printf("enter %d elements in list 2:\n",n);
  for(j=0;j<n;j++)
  {
    scanf("%d",&b[j]);
  }
  i=j=k=0;
  while(i<m&&j<n)
  {
    if(a[i]<b[j])
          c[k++]=a[i++];
```

```
      else
          c[k++]=b[j++];
   }
   while(i<m)
      c[k++]=a[i++];
   while(j<n)
      c[k++]=b[j++];
   printf("elements after merging:\n");
   for(k=0;k<m+n;k++)
   {
      printf("%d\n",c[k]);
   }
   return 0;
}
```

Output:



```
how many elements do you want to read into list 1:3
how many elements do you want to read into list 2:3
enter 3 elements in list 1:
12
15
27
enter 3 elements in list 2:
1
34
55
elements after merging:
1
12
15
27
34
55


=== Code Execution Successful ===
```

Inferences:

- Quick sort working principle is **Divide and Conquer** approach. It picks a **pivot** element, partitions the array into two halves (elements less than pivot & elements greater than pivot), and recursively sorts them.
- Quick Sort is **efficient, fast, and widely used in practice** with an average of **O(n log n)**. However, it is **not stable**, and poor pivot selection can lead to **O(n^2)** performance.
- Merge sort working principle is **Divide and Conquer** strategy. Recursively splits the array into halves until single elements remain, then **merges** them back in sorted order.
- Merge Sort is a **stable, guaranteed O(n log n)** sorting algorithm, excellent for **large datasets** and **linked lists**, but it needs **extra**

space (O(n)). Unlike Quick Sort, its performance doesn’t degrade to $O(n^2)$.