WEEK 6 ADA LAB

1BM21CS247

Q) Implement Merge Sort using a C program:-

SOURCE CODE:-

```
#include <stdio.h>

#define max 10

int a[30];
int b[10];

void merging(int low, int mid, int high) {
  int l1, l2, i;

  for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++) {
    if(a[l1] <= a[l2])
      b[i] = a[l1++];
    else
      b[i] = a[l2++];
  }

  while(l1 <= mid)
    b[i++] = a[l1++];

  while(l2 <= high)
    b[i++] = a[l2++];

  for(i = low; i <= high; i++)
    a[i] = b[i];
```

```
}

void sort(int low, int high) {
  int mid;

  if(low < high) {
    mid = (low + high) / 2;
    sort(low, mid);
    sort(mid+1, high);
    merging(low, mid, high);
  } else {
    return;
  }
}

int main() {
  int i,n;
  printf("Enter the number of elements to be sorted:\n");
  scanf("%d",&n);
  printf("Enter the list of elements to be sorted:\n");
  for(i=0;i<n;i++)
  {
     scanf("%d",&a[i]);
  }
  printf("List before sorting\n");

  for(i = 0; i < n; i++)
    printf("%d ", a[i]);
```

```
  sort(0, n);

  printf("\nList after sorting\n");

  for(i = 1; i <=n; i++)
    printf("%d ", a[i]);
}
```

OUTPUT:-

```
input
Enter the number of elements to be sorted:
10
Enter the list of elements to be sorted:
10 9 8 7 6 5 4 3 2 1
List before sorting
10 9 8 7 6 5 4 3 2 1
List after sorting
1 2 3 4 5 6 7 8 9 10

...Program finished with exit code 0
Press ENTER to exit console.
```

Q) Implement Quick sort using a C program

SOURCE CODE:-

```
#include <stdio.h>

// function to swap elements
void swap(int *a, int *b) {
  int t = *a;
  *a = *b;
  *b = t;
}
```

```
// function to find the partition position
int partition(int array[], int low, int high) {

  // select the rightmost element as pivot
  int pivot = array[high];

  // pointer for greater element
  int i = (low - 1);

  // traverse each element of the array
  // compare them with the pivot
  for (int j = low; j < high; j++) {
    if (array[j] <= pivot) {

      // if element smaller than pivot is found
      // swap it with the greater element pointed by i
      i++;

      // swap element at i with element at j
      swap(&array[i], &array[j]);
    }
  }

  // swap the pivot element with the greater element at i
  swap(&array[i + 1], &array[high]);

  // return the partition point
  return (i + 1);
}
```

```
void quickSort(int array[], int low, int high) {
  if (low < high) {

    // find the pivot element such that
    // elements smaller than pivot are on left of pivot
    // elements greater than pivot are on right of pivot
    int pi = partition(array, low, high);

    // recursive call on the left of pivot
    quickSort(array, low, pi - 1);

    // recursive call on the right of pivot
    quickSort(array, pi + 1, high);
  }
}

// function to print array elements
void printArray(int array[], int size) {
  for (int i = 0; i < size; ++i) {
    printf("%d  ", array[i]);
  }
  printf("\n");
}

// main function
int main() {
  int data[30];
  int n;
```

```
  printf("enter size of array:\n");
  scanf("%d", &n);
  printf("enter list of elements to be sorted: \n");
  for(int i=0; i<n; i++)
  {
     scanf("%d", &data[i]);
  }




  printf("Unsorted Array\n");
  printArray(data, n);

  // perform quicksort on data
  quickSort(data, 0, n - 1);

  printf("Sorted array in ascending order: \n");
  printArray(data, n);
}
```

OUTPUT:-

```
input
enter size of array:
10
enter list of elements to be sorted:
12 10 13 5 8 15 11 6 5 2
Unsorted Array
12  10  13  5  8  15  11  6  5  2
Sorted array in ascending order:
2  5  5  6  8  10  11  12  13  15


...Program finished with exit code 0
Press ENTER to exit console.
```