WEEK 15 - Program to perform Sorting

```
#include <stdio.h>
#include <stdlib.h>
void swap(int *a, int *b) {
int temp = *a;
*a = *b;
*b = temp;
}
int partition(int arr[], int low, int high) {
int pivot = arr[high];
int i = (low - 1);
for (int j = low; j <= high - 1; j++) {
if (arr[j] < pivot) {
i++;
swap(&arr[i], &arr[j]);
}
}
swap(&arr[i + 1], &arr[high]);
return (i + 1);
}
void quickSort(int arr[], int low, int high) {
if (low < high) {
int pi = partition(arr, low, high);
quickSort(arr, low, pi - 1);
quickSort(arr, pi + 1, high);
}
}
void merge(int arr[], int l, int m, int r) {
int i, j, k;
int n1 = m - l + 1;
int n2 = r - m;
```

```
int L[n1], R[n2];
for (i = 0; i < n1; i++)
L[i] = arr[l + i];
for (j = 0; j < n2; j++)
R[j] = arr[m + 1 + j];
i = 0;
j = 0;
k = l;
while (i < n1 && j < n2) {
if (L[i] <= R[j]) {
arr[k] = L[i];
i++;
} else {
arr[k] = R[j];
j++;
}
k++;
}
while (i < n1) {
arr[k] = L[i];
i++;
k++;
}
while (j < n2) {
arr[k] = R[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);
}
}
int main() {
int n;
printf("Enter the number of elements: ");
scanf("%d", &n);
int arr[n];
printf("Enter %d elements:\n", n);
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}
printf("\nSorting using Quick Sort:\n");
quickSort(arr, 0, n - 1);
for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
printf("\n\nSorting using Merge Sort:\n");
mergeSort(arr, 0, n - 1);
for (int i = 0; i < n; i++) {
printf("%d ", arr[i]);
}
return 0;
}
```

OUTPUT

```
Enter the number of elements: 3
Enter 3 elements:
123
```

145

639

Sorting using Quick Sort:

123 145 639

Sorting using Merge Sort:

123 145 639