LAB ASSIGNMENT 6
Sorting algorithms

1. Write a program to implement bubble sort for sorting n elements in an array.

2. Write a program to implement Selection sort for sorting n elements in an array.

3. Write a program to implement Insertion sort for sorting n elements in an array.

4. Write a program to implement Merge sort.

5. Write a program to implement Quick sort.

Solutions:

1. Write a program to implement bubble sort for sorting n

elements in an array. #include <stdio.h>

```
void bubbleSort(int a[],
  int n) { int i, j, temp;
  for(i = 0; i < n - 1; i++) {
    for(j = 0; j < n - i - 1; j++) {
      if(a[j] > a[j +
        1]) { temp =
        a[j];
        a[j] = a[j +
        1]; a[j + 1]
        = temp;
      }
    }
  }
}

int main() {
```

```
int n, i, a[100];
scanf("%d", &n);
for(i = 0; i < n; i++) scanf("%d", &a[i]);
```

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

2. Write a program to implement Selection sort for sorting n

elements in an array. #include <stdio.h>

```
void selectionSort(int a[],
  int n) { int i, j, min,
  temp;
  for(i = 0; i < n - 1;
    i++) { min = i;
    for(j = i + 1; j < n;
      j++) { if(a[j] <
      a[min])
        min = j;
    }
    temp =
    a[i]; a[i] =
    a[min];
    a[min] =
    temp;
  }
}

int main() {
  int n, i, a[100];
  scanf("%d", &n);
  for(i = 0; i < n; i++) scanf("%d",
  &a[i]); selectionSort(a, n);
  for(i = 0; i < n; i++) printf("%d ",
  a[i]); return 0;
}
```

3. Write a program to implement Insertion sort for sorting n

elements in an array. #include <stdio.h>

```
void insertionSort(int a[], int n) {
  int i, j, key;
  for(i = 1; i < n;
    i++) { key =
    a[i];
    j = i - 1;
    while(j >= 0 && a[j] >
      key) { a[j + 1] =
      a[j];
      j--;
    }
    a[j + 1] = key;
  }
}

int main() {
  int n, i, a[100];
  scanf("%d", &n);
  for(i = 0; i < n; i++) scanf("%d",
  &a[i]); insertionSort(a, n);
  for(i = 0; i < n; i++) printf("%d ",
  a[i]); return 0;
}
```

4. Write a program to implement

Merge sort. #include <stdio.h>

```
void merge(int a[], int l, int
  m, int r) { int n1 = m - l +
  1, n2 = r - m;
  int L[100], R[100];
  for(int i = 0; i < n1; i++) L[i] =
  a[l + i]; for(int j = 0; j < n2; j++)
  R[j] = a[m + 1 + j]; int i = 0, j =
  0, k = l;
  while(i < n1 && j < n2) {
    if(L[i] <= R[j]) a[k++] = L[i++];
    else a[k++] = R[j++];
  }
```

```
while(i < n1) a[k++] =
L[i++]; while(j < n2)
a[k++] = R[j++];
```

```
}

void mergeSort(int a[], int l,
   int r) { if(l < r) {
      int m = (l + r) / 2;
      mergeSort(a, l, m);
      mergeSort(a, m + 1,
      r); merge(a, l, m, r);
   }
}

int main() {
   int n, i, a[100];
   scanf("%d", &n);
   for(i = 0; i < n; i++) scanf("%d",
   &a[i]); mergeSort(a, 0, n - 1);
   for(i = 0; i < n; i++) printf("%d ",
   a[i]); return 0;
}
```

5. Write a program to implement

Quick sort. #include <stdio.h>

```
void swap(int *a, int
   *b) { int t = *a;
   *a = *b;
   *b = t;
}

int partition(int a[], int low, int
   high) { int pivot = a[high];
   int i = (low - 1);
   for(int j = low; j < high;
      j++) { if(a[j] < pivot)
      {
         i++;
         swap(&a[i], &a[j]);
```

```
        }
    }
    swap(&a[i + 1],
    &a[high]); return (i +
    1);
}

void quickSort(int a[], int low, int
    high) { if(low < high) {
        int pi = partition(a, low,
        high); quickSort(a, low,
        pi - 1); quickSort(a, pi
        + 1, high);
    }
}

int main() {
    int n, i, a[100];
    scanf("%d", &n);
    for(i = 0; i < n; i++) scanf("%d",
    &a[i]); quickSort(a, 0, n - 1);
    for(i = 0; i < n; i++) printf("%d ",
    a[i]); return 0;
}
```