COMPUTER LAB

NAME: SOUMYADIP MAITY

ROLL NO.: 22053029

SEC: CSE-49

YEAR: 2023-24

ASSIGNMENT – 13

1. **WAP to sort a set of numbers in ascending order using insertion sort, bubble sort, selection sort, quick sort, merge sort, and heap sort.**

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

// function to perform insertion sort
void insertionSort(int arr[ ], int n) {
  int i, j, temp;

  for (i = 1; i < n; i++) {
    temp = arr[i];
    j = i - 1;
    while (j >= 0 && arr[j] > temp) {
      arr[j + 1] = arr[j];
      j--;
    }
    arr[j + 1] = temp;
  }
}

// function to perform bubble sort
void bubbleSort(int arr[ ], int n) {
  int i, j, temp;

  for (i = 0; i < n; i++) {
    for (j = 0; j < n - i - 1; j++) {
      if (arr[j] > arr[j + 1]) {
        temp = arr[j];
        arr[j] = arr[j + 1];
        arr[j + 1] = temp;
      }
    }
  }
}

// function to perform selection sort
void selectionSort(int arr[ ], int n) {
  int i, min_idx, temp;

  for (i = 0; i < n; i++) {
    min_idx = i;
    for (int j = i + 1; j < n; j++) {
      if (arr[j] < arr[min_idx]) {
        min_idx = j;
```

```
      }
    }
    if (min_idx != i) {
      swap(arr[i], arr[min_idx]);
    }
  }
}

// function to perform quick sort
void quickSort(int arr[ ], int n) {
  int pivot, i, j, temp;

  if (n <= 1) {
    return;
  }

  // select a pivot element
  pivot = arr[n / 2];

  // partition the list around the pivot
  for (i = 0; i < n; i++) {
    if (arr[i] <= pivot) {
      j = i;
      while (j < n && arr[j] <= pivot) {
        j++;
      }
      swap(arr[j], arr[i]);
    }
  }

  // recursively sort the sublists
  quickSort(arr, n - i - 1);
  quickSort(arr + i, n - i - 1);
}

// function to perform merge sort
void mergeSort(int arr[ ], int n) {
  int mid = n / 2;
  int left = 0;
  int right = mid;
  int i, j, k;

  // merge the two sublists
  while (left < right) {
    k = left + right - 1;
    while (left < k && arr[left] <= arr[k]) {
```

```
      left++;
    }
    while (k > left && arr[k] <= arr[left]) {
      swap(arr[left], arr[k]);
      left++;
      k--;
    }
  }
}

// function to perform heap sort
void heapSort(int arr[ ], int n) {
  int i, j, max_idx, temp;

  // build a heap
  for (i = (n - 1) / 2; i > 0; i--) {
    max_idx = i;
    for (j = i * 2 - 1; j < n && arr[j] < arr[max_idx]; j++) {
      if (arr[j] > arr[max_idx]) {
        temp = arr[j];
        arr[j] = arr[max_idx];
        arr[max_idx] = temp;
      }
    }
  }

  // pop the heap until there are no more elements
  while (n > 0) {
    swap(arr[0], arr[-1]);
    n--;
  }
}
```