Practical-6

Aim: Implement selection sort, bubble sort, insertion sort, and heap sort using an array as a data structure.

Code:

```
#include <stdio.h>
void selectionSort(int arr[], int n);
void bubbleSort(int arr[], int n);
void insertionSort(int arr[], int n);
void heapSort(int arr[], int n);
void heapify(int arr[], int n, int i);
void printArray(int arr[], int n);

int main() {
   int arr[] = {64, 34, 25, 12, 22, 11, 90};
   int n = sizeof(arr) / sizeof(arr[0]);
   int choice;

   printf("Original array: \n");
   printArray(arr, n);

   printf("\nSelect Sorting Algorithm:\n");
   printf("1. Selection Sort\n");
   printf("2. Bubble Sort\n");
   printf("3. Insertion Sort\n");
   printf("4. Heap Sort\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         selectionSort(arr, n);
         printf("Sorted array using Selection Sort: \n");
         break;
      case 2:
         bubbleSort(arr, n);
         printf("Sorted array using Bubble Sort: \n");
         break;
      case 3:
         insertionSort(arr, n);
         printf("Sorted array using Insertion Sort: \n");
         break;
      case 4:
         heapSort(arr, n);
         printf("Sorted array using Heap Sort: \n");
         break;
      default:
         printf("Invalid choice!\n");
         return 1;
   }

   printArray(arr, n);
   return 0;
}
```

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

void insertionSort(int arr[], int n) {
   int i, key, j;
   for (i = 1; i < n; i++) {
      key = arr[i];
      j = i - 1;

      while (j >= 0 && arr[j] > key) {
         arr[j + 1] = arr[j];
         j = j - 1;
      }
      arr[j + 1] = key;
   }
}
void heapSort(int arr[], int n) {
   for (int i = n / 2 - 1; i >= 0; i--)
      heapify(arr, n, i);

   for (int i = n - 1; i >= 0; i--) {
      int temp = arr[0];
      arr[0] = arr[i];
      arr[i] = temp;

      heapify(arr, i, 0);
   }
}
void heapify(int arr[], int n, int i) {
```

```
    int largest = i;  // Initialize largest as root
    int left = 2 * i + 1;  // left child
    int right = 2 * i + 2;  // right child

    if (left < n && arr[left] > arr[largest])
        largest = left;
    if (right < n && arr[right] > arr[largest])
        largest = right;

    if (largest != i) {
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr, n, largest);
    }
}
void printArray(int arr[], int n) {
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

Output: