NAME – AADITA GARG
ROLL NO – 1024030461
SUB GROUP – 2C32

DATA STRUCTURES AND ALGORITHMS
ASSIGNMENT – 07 - SORTING

Write a program to implement following sorting techniques:

```
#include <iostream>
using namespace std;
void printArray(int arr[], int n) {
for (int i = 0; i < n; i++)
cout << arr[i] << " ";
cout << endl;
}
void selectionSort(int arr[], int n) {
for (int i = 0; i < n - 1; i++) {
int minIndex = i;
for (int j = i + 1; j < n; j++)
if (arr[j] < arr[minIndex])
minIndex = j;
swap(arr[i], arr[minIndex]);
}
}
void insertionSort(int arr[], int n) {
for (int i = 1; i < n; i++) {
int key = arr[i];
int j = i - 1;
while (j >= 0 && arr[j] > key) {
arr[j + 1] = arr[j];
j--;
```

```
}
arr[j + 1] = key;
}
}
void bubbleSort(int arr[], int n) {
for (int i = 0; i < n - 1; i++) {
bool swapped = false;
for (int j = 0; j < n - i - 1; j++) {
if (arr[j] > arr[j + 1]) {
swap(arr[j], arr[j + 1]);
swapped = true;
}
}
if (!swapped) break;
}
}
void merge(int arr[], int l, int mid, int r) {
int n1 = mid - l + 1;
int n2 = r - mid;
int a[n1], b[n2];
for (int i = 0; i < n1; i++) a[i] = arr[l + i];
for (int i = 0; i < n2; i++) b[i] = arr[mid + 1 + i];
int i = 0, j = 0, k = l;
while (i < n1 && j < n2) {
if (a[i] <= b[j]) arr[k++] = a[i++];
else arr[k++] = b[j++];
}
while (i < n1) arr[k++] = a[i++];
while (j < n2) arr[k++] = b[j++];
}
void mergeSort(int arr[], int l, int r) {
if (l >= r) return;
int mid = l + (r - l) / 2;
mergeSort(arr, l, mid);
mergeSort(arr, mid + 1, r);
```

```
merge(arr, l, mid, r);
}
int partition(int arr[], int l, int r) {
int pivot = arr[r];
int i = l - 1;
for (int j = l; j < r; j++) {
if (arr[j] <= pivot) {
i++;
swap(arr[i], arr[j]);
}
}
swap(arr[i + 1], arr[r]);
return i + 1;
}
void quickSort(int arr[], int l, int r) {
if (l < r) {
int pi = partition(arr, l, r);
quickSort(arr, l, pi - 1);
quickSort(arr, pi + 1, r);
}
}
int main() {
int arr[] = {64, 25, 12, 22, 11};
int n = 5;
cout << "Original Array: ";
printArray(arr, n);
selectionSort(arr, n);
// insertionSort(arr, n);
// bubbleSort(arr, n);
// mergeSort(arr, 0, n-1);
quickSort(arr, 0, n-1);
cout << "Sorted Array: ";
printArray(arr, n);
return 0;
}
```

```
Original Array: 64 25 12 22 11
Sorted Array: 11 12 22 25 64


=== Code Execution Successful ===
```

A slightly improved selection sort – We know that selection sort algorithm takes the minimum on every pass on the array, and place it at its correct position. The idea is to take also the maximum on every pass and place it at its correct position. So in every pass, we keep track of both maximum and minimum and array becomes sorted from both ends. Implement this logic.

```
#include <iostream>
using namespace std;

void improvedSelectionSort(int arr[], int n) {
   int left = 0;
   int right = n - 1;

   while (left < right) {
      int minIndex = left;
      int maxIndex = right;

      if (arr[minIndex] > arr[maxIndex])
         swap(arr[minIndex], arr[maxIndex]);

      for (int i = left + 1; i < right; i++) {
         if (arr[i] < arr[minIndex])
            minIndex = i;
         if (arr[i] > arr[maxIndex])
            maxIndex = i;
      }
```

```
        swap(arr[left], arr[minIndex]);

        if (maxIndex == left)
            maxIndex = minIndex;

        swap(arr[right], arr[maxIndex]);

        left++;
        right--;
    }
}

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];

    improvedSelectionSort(arr, n);

    cout << "Sorted Array: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
}
```

```
Enter number of elements: 5
Enter elements: 1 9 4 3 8
Sorted Array: 1 3 4 8 9
```