Data Structures (UCS301) Assignment

NAME-Suneet Arora **ROLL NO-1024030174**

```cpp
#include <bits/stdc++.h>
using namespace std;

void selectionSort(vector<int> &a) {
    int n = a.size();
    for(int i = 0; i < n-1; i++) {
        int minIndex = i;
for(int j = i+1; j < n; j++) { if(a[j]
         < a[minIndex]) minIndex = j;
        }
swap(a[i], a[minIndex]);
    }
}
```

```cpp
void insertionSort(vector<int> &a) {
    int n = a.size();
    for(int i = 1; i < n; i++) {
        int key = a[i];
int j = i - 1;

        while(j >= 0 && a[j] > key) {
            a[j+1] = a[j];
j--;
        }
a[j+1] = key;
    }
}
```

```
void bubbleSort(vector<int> &a) {
    int n = a.size();
    for(int i = 0; i < n-1; i++) {
        bool swapped = false;
        for(int j = 0; j < n-i-1; j++) {
            if(a[j] > a[j+1]) {
swap(a[j], a[j+1]); swapped = true;
            }
        }
if(!swapped) break;
    }
}
```

```
void merge(vector<int> &a, int l, int m, int r) {
    vector<int> left(a.begin()+l, a.begin()+m+1);
    vector<int> right(a.begin()+m+1, a.begin()+r+1);

int i=0, j=0, k=l;
    while(i < left.size() && j < right.size()) {
        if(left[i] <= right[j]) a[k++] = left[i++];
        else a[k++] = right[j++];
    }
while(i < left.size()) a[k++] = left[i++];
    while(j < right.size()) a[k++] = right[j++];
}

void mergeSort(vector<int> &a, int l, int r) {
    if(l >= r) return;
    int m = (l + r) / 2;
    mergeSort(a, l, m);
    mergeSort(a, m+1, r);
    merge(a, l, m, r);
}
```

```
int partition(vector<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(vector<int> &a, int low, int high) {
    if(low < high) {
        int p = partition(a, low, high);
        quickSort(a, low, p-1);
        quickSort(a, p+1, high);
    }
}
```

```
void improvedSelectionSort(vector<int> &a) {
    int left = 0;
int right = a.size() - 1;

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

        // Important: if the left element is > right element swap roles
if(a[minIndex] > a[maxIndex])
swap(minIndex, maxIndex);

        // scan between left and right
for(int i = left + 1; i <= right - 1; i++) {
```

```
            if(a[i] < a[minIndex])
                minIndex = i;
            else if(a[i] > a[maxIndex])
                maxIndex = i;
        }

        // place minimum at left
        swap(a[left], a[minIndex]);

        // if max element was at left, it moved to minIndex after the previous
swap
        if(maxIndex == left)
            maxIndex = minIndex;

        // place maximum at right
        swap(a[right], a[maxIndex]);

        left++;
        right--;
    }
}
```