Practical - 1

AIM: Write programs to implement the following
implement following algo. and print number of comparison and swaps/shifts for each of the following cases.
case 1: input data is already sorted
case 2: input data is reversely sorted
case 3: input data is randomly ordered a> length ~~even~~ odd
b> length ~~odd~~ even

Algorithm 1: Insertion Sort

Algo: mark first element as sorted
for each unsorted element x
extract the element x
for J ← lastunsorted index down to 0
if current element J > x
move sorted element to right by 1
else break the loop and insert x there

Code:

```
Void insertionSort(vector<int> &arr) {
    int i, key, J, n = arr.size();
    int countComp = 0;   // Counting Comparisons
    int countSwap = 0;   // Counting Swap
    for (int i=0; i<n; i++) {
        key = arr[i]
        J = i-1;
        CountComp++;   // first while loop comp check
```

```
while (j >= 0 && arr[j] > key) {
    CountComp++;
    CountSwap++;   // every time when
                   // condition is true swap
                   // take a place
    arr[j+1] = arr[j];
    j--;
}
arr[j+1] = key;
}
Cout << "Total numbers of comparisons are" <<
    CountComp
<< "\n Total numbers of swaps are" <<
    CountSwap << endl;
}
```

Case 1: 7, 9, 12, 13, 17

Output:

Total numbers of comparisons are 4 and
Total numbers of swaps are 0.

Case 2: 17, 13, 12, 9, 7

Total numbers of comparisons are 14 and
Total numbers of swaps are 10.

Case 3: 7, 12, 13, 9, 17

Total numbers of comparisons are 6 and
total numbers of swaps are 2

b) 7, 12, 13, 9, 17, 1

Total Numbers of comparisons are 12 and

Total Numbers of swaps are 7.

Algo 2: Selection Sort

Algo: repeat (numberOfElement - 1) times
    Set first unsorted element as minimum
    for each unsorted elements
        if (element < current Minimum)
            Set new element as minimum
    Swap minimum with first unsorted position.

Code:
```
Void selectionSort (vector<int> & arr) {
    int i, J, minIndex, n = arr.size();
    int CountComp = 0;
    int CountSwap = 0;
    for (i=0; i<n-1; i++) {
        minIndex = i;
        for (J = i+1; J<n; J++) {
            CountComp++;
            if (arr[J] < arr[minIdx])
                minIdx = J;
        }
        // CountComp++; we not considering this comp.
        if (minIdx != i) {          // because it not hoping
            Swap(arr[minIdx, arr[i]); // b/w arrays elements.
            CountSwap++;
        }
    }
}
```

Case 1: 7, 9, 12, 13, 17

Total numbers of comparisons are 10

```
cout << "Total numbers of Comparisons are"
     << countComp <<
     << "\n Total numbers of Swaps are"
     << countSwap << endl;
}
```

Case 1: 7, 9, 12, 13, 17
Total numbers of comparisons are 10
Total numbers of swaps are 0.

Case 2: 17, 13, 12, 9, 7
Total numbers of comparisons are 10
Total numbers of swaps are 2.

Case 3: 7, 12, 13, 9, 17
Total numbers of comparisons are 10
Total numbers of ~~comparisons~~ swaps are 1.

Case 4: 7, 12, 13, 9, 17, 1
Total numbers of comparisons are 15
Total numbers of ~~comparis~~ swaps are 5

Sorting method	ascending order.		descending order.		Random order.	
	Comp	Swap	Comp	Swap	Comp	Swap
Insertion sort	4	0	14	10	6	2
	O(n)		O(n²)		O(n²)	
Selection sort	10	0	10	2	10	2
	O(n²)		O(n²)		O(n²)	

↳ for selection sort theire are same number of comp. for every case and for insertion sort best case is when array is sorted so lowest number of Comp are there but worst case is when array is Revessly sorted so highest number of comp. are this.

* Magic Square

Cond 1: position of next number by decrementing row number. the prev. number by 1, incrementing col number by 1.

if row = -1 → n-1

mcol = n → 0

Cond 2: if the magic sq. already contains a Number at the calculated Col. position will be dec by 2, and calculated row position will be inc. by 1.

Cond 3: if the calculated row positions is -1 & Calculated column position is n, the new position would be: (0, n-2)

Code:

```
Void MagicSquare(int n) {
int squ[n][n];
int i,j, num;
for(i=0; i<n; i++) {
    for(j=0; j<n; j++) {
        magicsqu[i][j] = 0;
    i=n/2;
    j=n-1;
}
for(num=1; num<=n*n;) {
    if(i==-1 && j==n) {// 3rd cond.
        j=n-2;
        i=0;
    }
    else {
        if(j==n)
            j=0
        if(i<0)
            i=n-1;
    }
    if(magicsqu[i][j]) {
        j-=2;
        i++;
        Continue;
    }
    else
        squ[i][j] = num++;
```

```
        j++;
        i--;
    }

    for(i=0; i<n; i++) {
        for(j=0; j<n; j++)
            Printf("%d", sqa[i][j]);
        Printf("\n");
    }
}

Void main() {
    int n=3;    // for only odd numbers
    squ(n);
}
```

23/12/22