# PROGRAM – 04

## PART – 4 A

**Aim - WAP to implement a Insertion Sort.**

### ❖ Algorithm

```
Insertion-Sort(A)
for j = 2 to A.length          ----------------------------------------------   (n+1) unit time
    key = A[j]                 ----------------------------------------------   1 unit time
    i = j – 1                  ----------------------------------------------   1 unit time
    while i > 0 and A[i] > key ------------------------------------------------ (n) unit time
        A[i + 1] = A[i]        ----------------------------------------------   1 unit time
        i = i – 1              ----------------------------------------------   1 unit time
        A[i + 1] = key         ----------------------------------------------   1 unit time
```

### ❖ Time Complexity

$T(n) = O(n^2)$ => Best and Average Case.

$T(n) = O(n)$ => Best Case.

### ❖ Space Complexity

$S(n) = O(n)$

### ❖ Source Code

```
#include <stdio.h>
void insertionSort(int A[], int n) {
   int i, j, key;
   for (j = 1; j < n; j++) {
      key = A[j];
      i = j - 1;
      while (i >= 0 && A[i] > key) {
         A[i + 1] = A[i];
         i = i - 1;
      }
      A[i + 1] = key;
   }
```

```
}


void printArray(int A[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", A[i]);
    }
    printf("\n");
}
int main() {
    int A[] = {12, 11, 13, 5, 6};
    int n = sizeof(A) / sizeof(A[0]);

    printf("Original array: ");
    printArray(A, n);

    insertionSort(A, n);

    printf("Sorted array: ");
    printArray(A, n);
    return 0;
}
```

❖ **Output :**

```
Original array: 12 11 13 5 6
Sorted array: 5 6 11 12 13
```

## PART – 4 B

## Aim - WAP to implement a Selection Sort.

### ❖ ALGORIHM

```
for(i=0; i<n; i++){          ----------   (n+1) unit time
   int min = 1;              ----------   1 unit time
   for(j=i+1; j<n; j++){     ----------   (n) unit time
      if(a[j] < a[min]){     ----------   1 unit time
         min = j;            ----------   1 unit time
      }
      if(min != i){          ----------   1 unit time
         swap(a[i],a[min]);  ----------   1 unit time
      }
   }
}
```

### ❖ Time Complexity

$T(n) = O(n^2)$

### ❖ Space Complexity

$S(n) = O(n)$

### ❖ Source Code

```
#include <stdio.h>
void selectionSort(int A[], int n) {
   int i, j, min;
   for (i = 0; i < n; i++) {
      min = i;
      for (j = i + 1; j < n; j++) {
         if (A[j] < A[min]) {
            min = j;
         }
```

```
            }
            if (min != i) {
                int temp = A[i];
                A[i] = A[min];
                A[min] = temp;
            }
        }
    }
    void printArray(int A[], int n) {
        for (int i = 0; i < n; i++) {
            printf("%d ", A[i]);
        }
        printf("\n");
    }
    int main() {
        int A[] = {29, 10, 14, 37, 13};
        int n = sizeof(A) / sizeof(A[0]);
        printf("Original array: ");
        printArray(A, n);
        selectionSort(A, n);
        printf("Sorted array: ");
        printArray(A, n);
        return 0;
    }
```

❖ **Output :**

```
Original array: 29 10 14 37 13
Sorted array: 10 13 14 29 37
```