PART A

(PART A : TO BE REFFERED BY STUDENTS)

Experiment No.01

A.1 Aim:

Write a program to implement Selection Sort & Insertion sort and analyze its complexity.

A.2 Prerequisite:

A.3 Outcome:

After successful completion of this experiment students will be able to analyze the time complexity of various classic problems.

A.4 Theory:

Selection sort is a sorting algorithm, specifically an in-placecomparison sort. It has O(n^2) time complexity, making it inefficient on large lists, and generally performs worse than the similar insertion sort. Selection sort is noted for its simplicity, and it has performance advantages over more complicated algorithms in certain situations, particularly where auxiliary memory is limited.The algorithm divides the input list into two parts: the sublist of items already sorted, which is built up from left to right at the front (left) of the list, and the sublist of items remaining to be sorted that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted subsist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

Algorithm:

```
Start
for i = 1:n,
    k = i
    for j = i+1:n,
        if a[j] < a[k], k = j
    → invariant: a[k]
         smallest of a[i..n]
swap a[i,k]
    → invariant: a[1..i] in final position
end
```

Time Complexity:

The number of iterations of this loop is nnn in the first call, then n−1n-1n−1, then n−2n-2n−2, and so on. We've seen that this sum, 1+2+⋯+(n−1)+n1 + 2 + \cdots + (n-1) + n1+2+⋯+(n−1)+n is an arithmetic series, and it evaluates to (n+1)(n/2)(n+1)(n/2)(n+1)(n/2), or n2/2+n/2n^2/2 + n/2n2/2+n/2. Therefore, the total time for all calls to indexOfMinimum is some constant times n2/2+n/2n^2/2 + n/2n2/2+n/2. In terms of big-Θ notation, we don't care about that constant factor, nor do we care about the factor of 1/2 or the low-order term. The result is that the running time for all the calls to indexOfMinimum is Θ(n2)\Theta(n^2)Θ(n2).

PART B

(PART B : TO BE COMPLETED BY STUDENTS)

Roll No.: 5	Name: Tavishaa Jaiswal
Class: SE(COMP)	Batch: C1
Date of Experiment:	Date of Submission
Grade:	

B.1 Software Code written by student:

Insertion Sort:

```
#include<stdio.h>
#include<conio.h>
#define size 100
int arr[size];
int c=0;
void Sort(int a[],int n)
{
  int i,j,key;
  for(i=1,c++;c++,i<n;i++,c++)
  {
    key=a[i];  c++;
    j=i-1;  c++;
    while((j>=0)&&(a[j]>key))
    {
      c++;
      a[j+1]=a[j]; c++;
      j=j-1; c++;
    }
    a[j+1]=key; c++;
  }
  printf("\nSORTED LIST : \n"); c++;
  for(i=0,c++;c++,i<n;i++,c++)
  {
    printf("%d ",a[i]);  c++;
  }
}
void main()
{
  int i,n;
  clrscr();
  printf("\nENTER SIZE : "); c++;
  scanf("%d",&n); c++;
  printf("\nENTER ELEMENTS : \n"); c++;
  for(i=0,c++;c++,i<n;i++,c++)
  {
    scanf("%d",&arr[i]);  c++;
  }
  printf("\nUNSORTED LIST : \n"); c++;
  for(i=0, c++;c++, i<n;i++, c++)
  {
    printf("%d ",arr[i]);  c++;
  }
  Sort(arr,n);
  printf("\nComplexity = %d",c);
  getch();
}
```

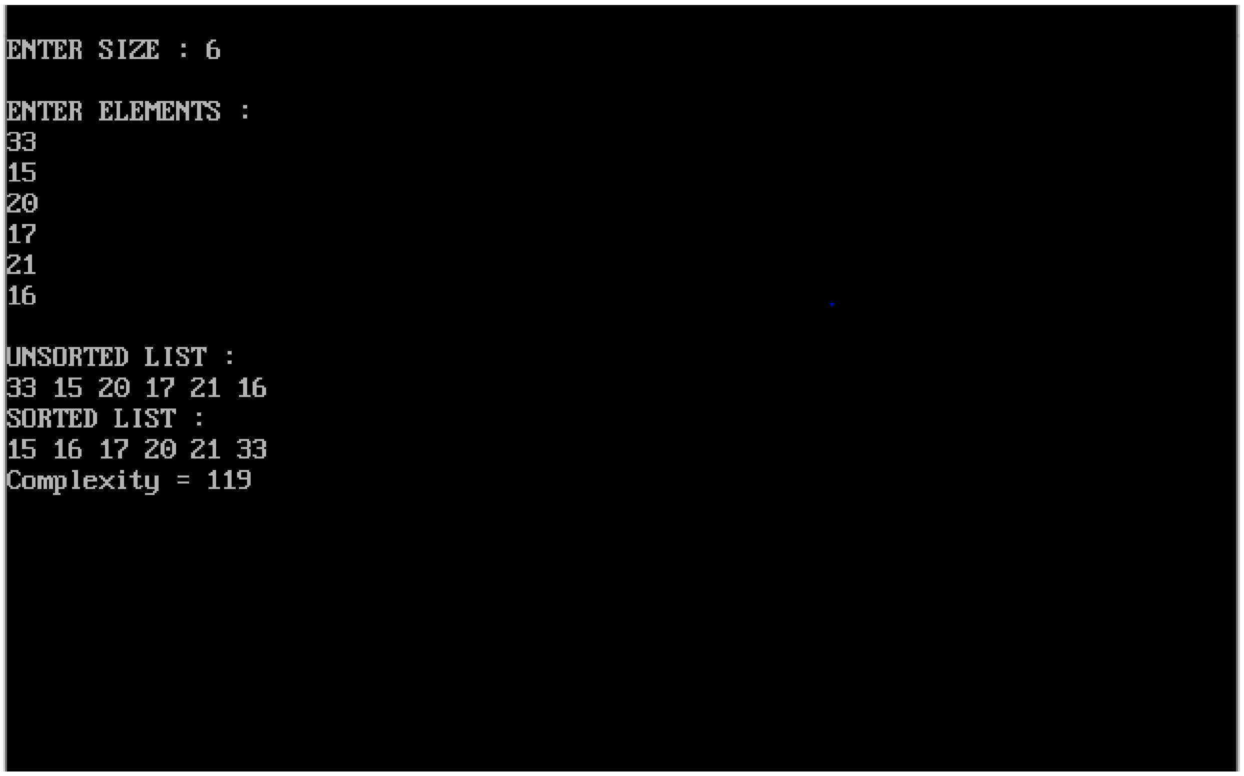
Selection Sort:

```
#include<conio.h>
#include<stdio.h>
#define size 100
int arr[size],c=0;
void SelSort(int a[],int n)
{
  int min,p,i,j;
  for(i=0,c++;c++,i<n-1;i++,c++)
  {
    min=a[i];  c++;
    p=i;  c++;
    for(j=i+1,c++;c++,j<n;j++,c++)
    {
      if(a[j]<min)
      { c++;
            min=a[j];  c++;
            p=j;  c++;
      }
    }
    min=a[p];  c++;
    a[p]=a[i];   c++;
    a[i]=min;  c++;
  }
  printf("\nSORTED ARRAY : \n");  c++;
  for(i=0,c++;c++,i<n;i++,c++)
  {
    printf("%d ",a[i]);  c++;
  }
}
void main()
```

```
{
  int i,n;
  clrscr();
  printf("\nENTER SIZE : ");  c++;
  scanf("%d",&n);  c++;
  printf("\nENTER ELEMENTS : \n"); c++;
  for(i=0,c++;c++,i<n;i++,c++)
  {
    scanf("%d",&arr[i]);  c++;
  }
  printf("\nUNSORTED ARRAY : \n");  c++;
  for(i=0,c++;c++,i<n;i++,c++)
  {
    printf("%d ",arr[i]);  c++;
  }
  SelSort(arr,n);
  printf("\nComplexity = %d",c);
  getch();
}
```

B.2 Input and Output:

Insertion Sort:



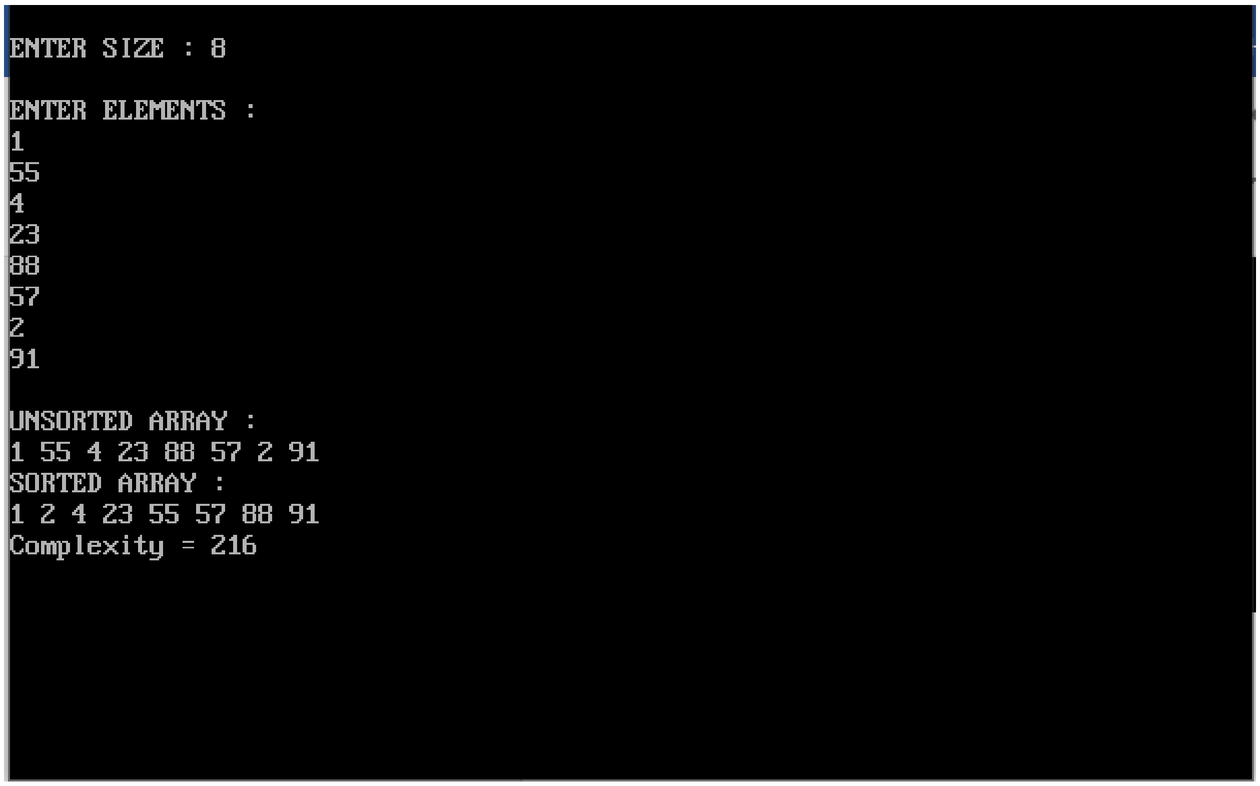
```

ENTER SIZE : 6

ENTER ELEMENTS :
33
15
20
17
21
16

UNSORTED LIST :
33 15 20 17 21 16
SORTED LIST :
15 16 17 20 21 33
Complexity = 119
```

Selection Sort:



```
ENTER SIZE : 8

ENTER ELEMENTS :
1
55
4
23
88
57
2
91

UNSORTED ARRAY :
1 55 4 23 88 57 2 91
SORTED ARRAY :
1 2 4 23 55 57 88 91
Complexity = 216
```

B.3 Observations and learning:

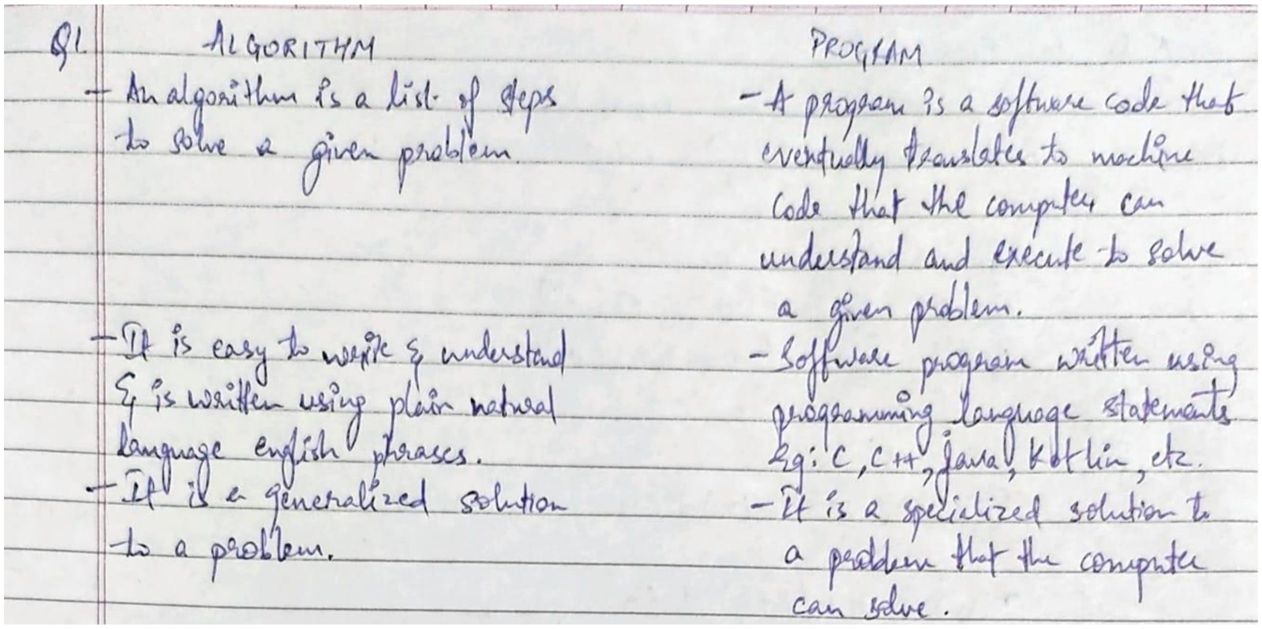
Insertion sort is a simple sorting algorithm that builds the final sorted list by transferring one element at a time. Selection sort, in contrast, is a simple sorting algorithm that repeatedly searches remaining items to find the smallest element and moves it to the correct location. Insertion sort is more complex than selection sort. We have successfully implemented selection sort and insertion sort using C language.

B.4 Conclusion:

Implemented Selection Sort and Insertion sort and analyzed its complexity.

B.5 Question of Curiosity

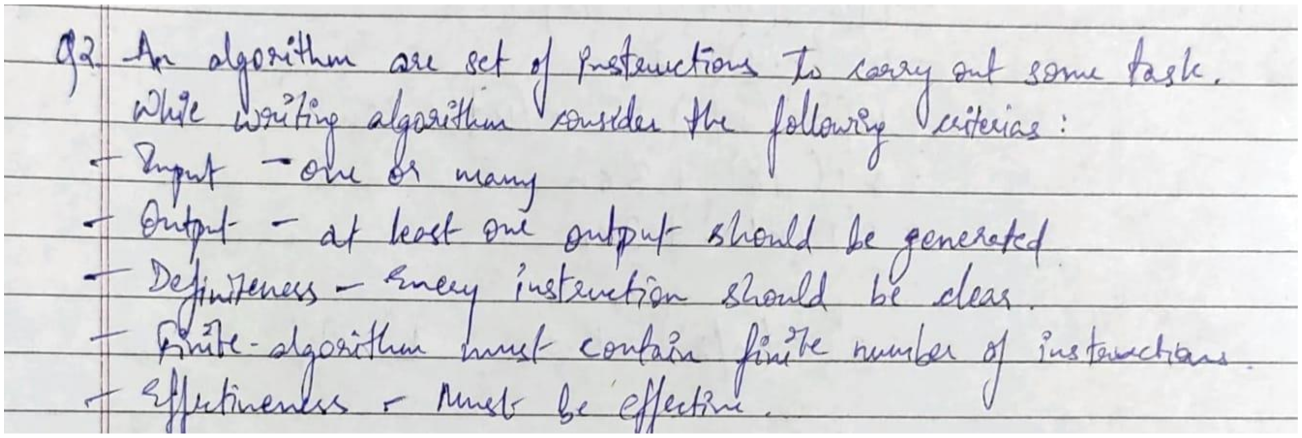
Q1: What is difference between algorithm and program?



Q1.

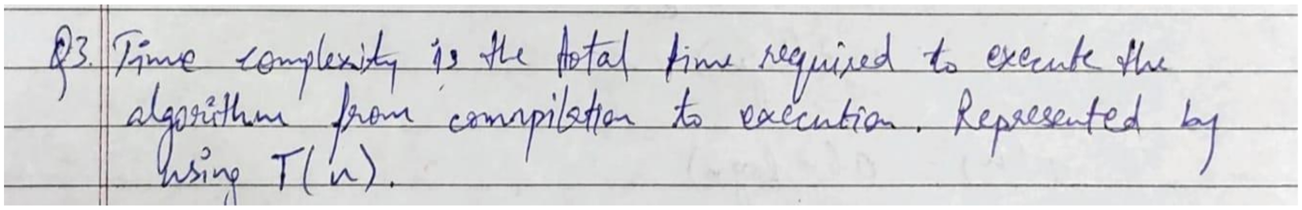
ALGORITHM	PROGRAM
- An algorithm is a list of steps to solve a given problem.	- A program is a software code that eventually translates to machine code that the computer can understand and execute to solve a given problem.
- It is easy to write & understand & is written using plain natural language english phrases.	- Software program written using programming language statements Eg: C, C++, Java, Kotlin, etc.
- It is a generalized solution to a problem.	- It is a specialized solution to a problem that the computer can solve.

Q2: What are different criteria to write algorithm?



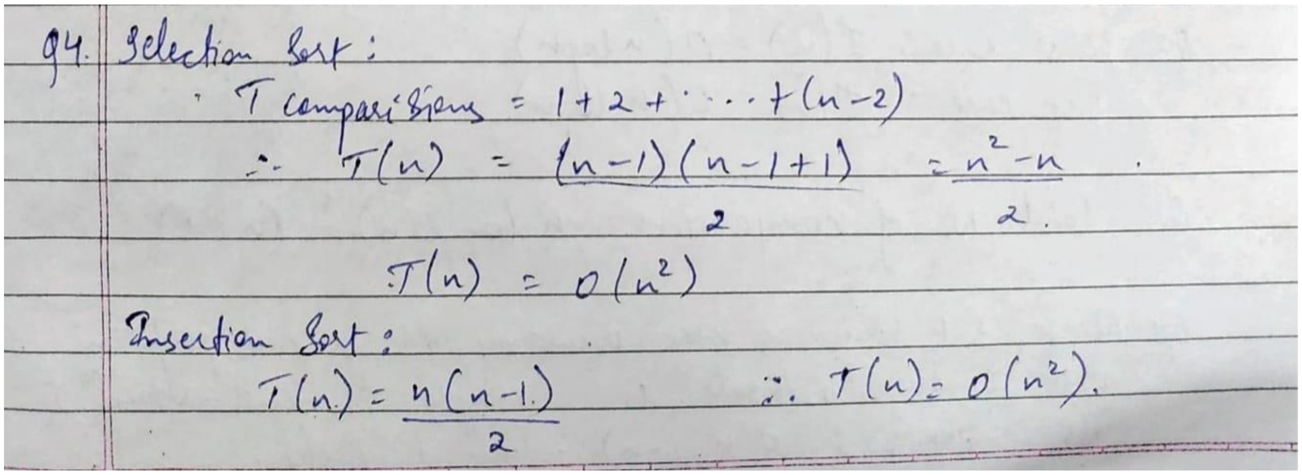
Q2. An algorithm are set of Instructions to carry out some task. While writing algorithm consider the following criterias:
- Input - one or many
- Output - at least one output should be generated
- Definiteness - Every instruction should be clear.
- Finite - algorithm must contain finite number of instructions.
- Effectiveness - Must be effective.

Q3: What is time complexity?



Q3. Time complexity is the total time required to execute the algorithm from compilation to execution. Represented by using T(n).

Q4: Derive time complexity if selection sort & Insertion Sort?



Q4. Selection Sort :

$\because$ T comparisions $= 1 + 2 + \cdots + (n-2)$

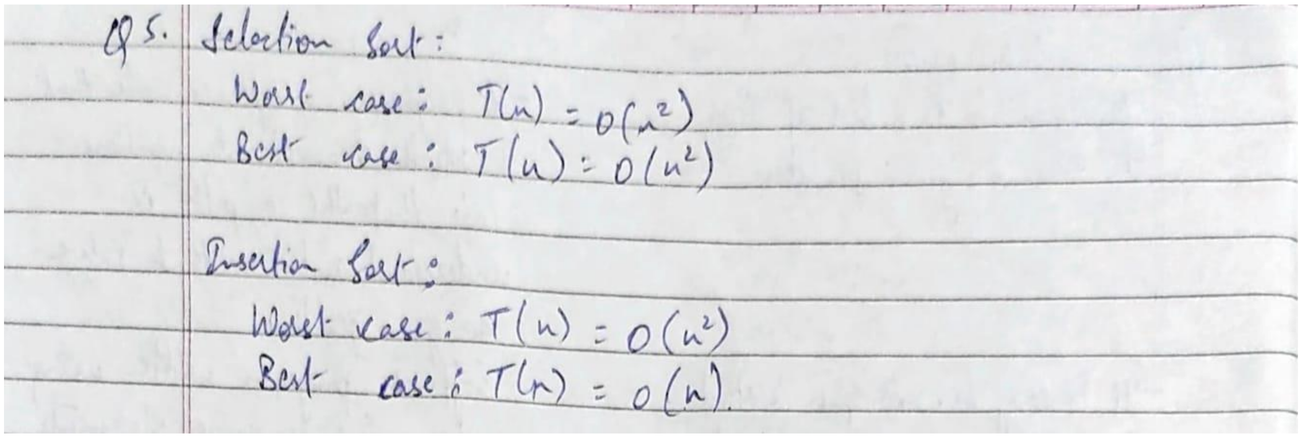
$\therefore \; T(n) = \frac{(n-1)(n-1+1)}{2} = \frac{n^2 - n}{2}.$

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

Insertion Sort :

$T(n) = \frac{n(n-1)}{2} \qquad \therefore \; T(n) = O(n^2).$

Q5: What is worst case and best case time complexity of selection sort & Insertion Sort?



Q5. Selection Sort :

Worst case : $T(n) = O(n^2)$

Best case : $T(n) = O(n^2)$

Insertion Sort :

Worst case : $T(n) = O(n^2)$

Best case : $T(n) = O(n).$