ASSIGNMENT - 2: Algorithms (Computation)

Problem Statement:

Consider the following computational problems. Find out the Time Complexity of all the problems. Write your comments and observations for each problem.

1. Find the sum of two numbers A and B

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’A’, ‘B’ and ‘Sum’
Step - 3: Read the values of ‘A’ and ‘B’ from the user / initialize them
Step - 4: Add ‘A’ and ‘B’. Assign the result to ‘Sum’
Step - 5: If required, print the value of ‘Sum’ to observe the output
Step - 6: End
```

Time Complexity: O(1)

Observation / Comments:
Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken for performing the addition of two numbers.

2. Convert temperature from Celsius(C) to Fahrenheit(F) and Fahrenheit(F) to Celsius(C)

Algorithm (Celsius to Fahrenheit):

```
Step - 1: Start
Step - 2: Declare variables ’C’ and ‘F’
Step - 3: Read the value of ‘C’ from the user / initialize it
Step - 4: Perform the calculation (9/5)*C + 32. Assign the result to ‘F’
Step - 5: If required, print the value of ‘F’ to observe the output
Step - 6: End
```

Algorithm (Fahrenheit to Celsius):

```
Step - 1: Start
Step - 2: Declare variables ’C’ and ‘F’
Step - 3: Read the value of ‘F’ from the user / initialize it
Step - 4: Perform the calculation (5/9)*(F - 32). Assign the result to ‘C’
Step - 5: If required, print the value of ‘C’ to observe the output
Step - 6: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken for performing Celcius to Fahrenheit and Fahrenheit to Celsius.

3. Find Area(A) and Perimeter(P) of a square

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’a’, ‘A’ and ‘P’
Step - 3: Read the value of ‘a’ (Length of one of its side) from the user / initialize it
Step - 4: Perform the calculation a*a. Assign the result to ‘A’
Step - 5: Perform the calculation 4*a. Assign the result to ‘P’
Step - 6: If required, print the value of ‘A’ and ‘P’ to observe the output
Step - 7: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken to find the Area and Perimeter of the square.

4. Find the Compound Interest (CI):

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’A’, ‘P’ , ‘r’, ‘n’ and ‘t’
Step - 3: Read the values of ‘P’ , ‘r’, ‘n’ and ‘t’ from the user / initialize it
Step - 4: Perform the calculation P*((1+r/n)^(nt)) and assign the result to A
Step - 5: If required, print the value of ‘A’ to observe the output
Step - 6: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken to find the value of Compound interest.

5. Swap two numbers using temporary variable

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’a’, ‘b’ and ‘temp’
Step - 3: Read the values of ‘a’ and ‘b’ from the user / initialize it
Step - 4: Store the value of ‘a’ in ‘temp’ (ie, perform temp = a)
Step - 5: Store the value of ‘b’ in ‘a’ (ie, perform a = b)
Step - 6: Store the value of ‘temp’ in ‘b’ (ie, perform b = temp)
Step - 7: If required, print the value of ‘a’ and b’ to observe the output
Step - 8: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken to swap two numbers.

6. Find the smallest of 2 numbers A and B

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’A’, ‘B’ and ‘Small’
Step - 3: Read the values of ‘A’ and ‘B’ from the user / initialize it
Step - 4: Check if ‘A’ is less than or equal to ‘B’ (ie, A<=B)
Step - 5: If it is, then assign Small = ‘A’. Otherwise assign Small = ‘B’
Step - 6: If required, print the value of ‘Small’ to observe the output
Step - 7: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken to find the smallest number.

7. Find the largest of three numbers A, B and C

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ’A’, ‘B’, ‘C’ and ‘Large’
Step - 3: Read the values of ‘A’ and ‘B’ from the user / initialize it
Step - 4: Check if ‘A’ is more than or equal to ‘B’ (ie, A>=B)
Step - 5: If it is, then assign Large = A. Otherwise assign Large = B
Step - 6: Check if ‘Large’ is more than or equal to ‘C’ (ie, Large>=C)
Step - 7: If it is, leave it as it was. Otherwise, assign Large = C
Step - 8: If required print the value of ‘Large’ to observe the output
Step - 9: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. Also there are no non-recursive and non-loop statements. Hence the time complexity is constant (O(1)), ie; Constant time is taken to find the largest number.

8. Find even numbers between 1 to 50

Algorithm:

```
Step - 1: Start
Step - 2: Declare variable ‘i’ and initialize it with 1. (ie, i = 1)
Step - 3: Check if ‘i’ is an even number ( i%2 == 0 )
Step - 4: If it is, print the value of ‘i’
Step - 5: Increment ‘i’ by one value.
Step - 6: Go to Step- 3 till the value of ‘i’ is 50
Step - 7: End
```

Time Complexity: O(1)

Observation / Comments:

Since the execution time of this algorithm is independent of the size of input, time taken to execute this remains the same. A loop or recursion that runs a constant number of times is considered as O(1). Thus, the time complexity is constant (O(1)), ie; Constant time is taken to find the even numbers between 1 and 50.

9. Find the sum of series 1+2+3+...+n

Algorithm:

```
Step - 1: Start
Step - 2: Declare variables ‘i’, ‘n’, ‘Sum’
Step - 3: Read the value of ‘n’ from user
Step - 4: Initialize ‘i’ with 1 and ‘Sum’ with 0. (ie, i = 1, Sum = 0)
Step - 5: Perform Sum = Sum + i
Step - 6: Increment ‘i’ by one value
Step - 7: Go to Step- 5 till the value of ‘i’ is n
Step - 8: If required, print the value of ‘Sum’ to observe the output
Step - 9: End
```

Time Complexity: O(n)

Observation / Comments:

Since the execution time of this algorithm is dependent on the size of input, time taken to execute is not constant. Time Complexity of a loop is considered as O(n) if the loop variable is incremented / decremented by a constant amount and same is the case when we want to find the sum of series.