Examples of Algorithm

Problem 1: Find the area of a Circle of radius r.

Inputs to the algorithm:
Radius r of the Circle.
Expected output:
Area of the Circle
Algorithm:

```
Step1: Start
Step2: Read\input the Radius r of the Circle
Step3: Area PI*r*r // calculation of area
Step4: Print Area
Step5: End
```

Problem2: Write an algorithm to read two numbers and find their sum.

Inputs to the algorithm:
First num1.
Second num2.
Expected output:
Sum of the two numbers.
Algorithm:

```
Step1: Start
Step2: Read\input the first num1.
Step3: Read\input the second num2.
Step4: Sum num1+num2 // calculation of sum
Step5: Print Sum
Step6: End
```

Problem 3: Convert temperature Fahrenheit to Celsius
Inputs to the algorithm:
Temperature in Fahrenheit
Expected output:
Temperature in Celsius
Algorithm:

```
Step1: Start
Step 2: Read Temperature in Fahrenheit F
Step 3: C 5/9*(F32)
Step 4: Print Temperature in Celsius: C
Step5: End
```

Type of Algorithms

The algorithm and flowchart, classification to the three types of *control Structures*. They are:

1. Sequence
2. Branching (Selection)
3. Loop (Repetition)

These three control structures are sufficient for all purposes. The sequence is exemplified by sequence of statements place one after the other – the one above or before another gets executed first. In flowcharts, sequence of statements is usually contained in the rectangular process box.

- The ***branch*** refers to a binary decision based on some condition. If the condition is true, one of the two branches is explored; if the condition is false, the other alternative is taken. This is usually represented by the 'if-then' construct in pseudo-codes and programs. In flowcharts, this is represented by the diamond-shaped decision box. This structure is also known as the *selection* structure.

Problem1: write algorithm to find the greater number between two numbers
Inputs to the algorithm:
First A.
Second B.
Expected output:
the greater number
Algorithm:

```
Step1: Start
Step2: Read/input A and B
Step3: If A greater than B then C=A
Step4: if B greater than A then C=B
Step5: Print C
Step6: End
```

 The *loop* allows a statement or a sequence of statements to be repeatedly executed based on some loop condition. It is represented by the ‘while’ and ‘for’ constructs in most programming languages, for unbounded loops and bounded loops respectively. (Unbounded loops refer to those whose number of iterations depends on the eventuality that the termination condition is satisfied; bounded loops refer to those whose number of iterations is known before-hand.) In the flowcharts, a back arrow hints the presence of a loop. A trip around the loop is known as iteration. You must ensure that the condition for the termination of the looping must be satisfied after some finite number of iterations, otherwise it ends up as an infinite loop, a common mistake made by inexperienced programmers. The loop is also known as the *repetition* structure.

Problem1: An algorithm to calculate even numbers between 0 and 99
Inputs to the algorithm:
Sequence of numbers
Expected output:
The calculate of even number
Algorithm:

```
1. Start
2. I ← 0
3. Write I in standard output
4. I ← I+2
5. If (I <=98) then go to line 3
6. End
```

Problem2: Design an algorithm which gets a natural value, n,as its input and calculates odd numbers equal or less than n. Then write them in the standard output:

```
1. Start
2. Read n
3. I ← 1
4. Write I
5. I ← I + 2
```

```
6. If ( I <= n) then go to line 4
7. End
```

Combining the use of these control structures, for example, a loop within a loop (nested loops), a branch within another branch (nested if), a branch within a loop, a loop within a branch,