Pseudo code assignments / examples

1.
Problem: Accept Grades from a Keyboard. When a negative grade comes the loop will stop. Calculate the average grade of the whole class.
Solution: The algorithm needs a LOOP. Since we don't know how often the loop will run beforehand (we don't know how many grades will be entered) we can not use the FOR-loop. WHILE or REPEAT-loop are fine.

```
Initialise: Grade = 0, Total-Grades = 0, Counter = 0
REPEAT
    READ Grade                                  (this is the input!)
    ADD Grade to Total-Grade
    ADD 1 to Counter        (need to count how many grades are entered!)
UNTIL   Grade < 0           (so go on until a grade is negative)
Average-Grade = Total-Grade / Counter
WRITE "Average grade is" Average-Grade          (this is the output!)
```

Here you can see the solution with a WHILE-loop. Remember: the WHILE-loop checks the ending condition of the loop **as** it starts.

```
Initialise: Grade = 0, Total-Grades = 0, Counter = 0
WHILE Grade >= 0
    ADD 1 to Counter        (need to count how many grades are entered!)
    READ Grade                                  (this is the input!)
    ADD Grade to Total-Grade
ENDWHILE
Average-Grade = Total-Grade / Counter
WRITE "Average grade is" Average-Grade          (this is the output!)
```

2.
Problem: Little game: Accept numbers thrown with 2 dice. Output the number of throws needed for a person to achieve a double SIX . (So: repeat throwing until you have a total of 12)
Solution: The algorithm needs a LOOP. Again we don't know how often , so we cannot use a FOR-loop. WHILE or REPEAT are both fine.

Here you can see the solution with a WHILE-loop. Remember: the WHILE-loop checks the ending condition of the loop **as (before)** it starts

```
Initialise: Counter = 0      (a variable, used to 'count' the number of throws)
            Total-of-Dice = 0
WHILE Total-of-Dice < 12        (if it is 12, it's a double 6 and we stop)
    ADD 1 to Counter        (need to count how many grades are entered!)
    READ Dice-1 , READ Dice-2          (this is the input!)
```

```
    Total-of-Dice = Dice-1 + Dice-2
ENDWHILE
WRITE "The number of throws is " Counter        (this is the output!)
```

Now you can see the solution with a REPEAT-loop. Remember: the REPEAT-loop checks the ending condition of the loop **after** it runs the first time.

```
Initialise:     Counter = 0        (used to 'count' the number of throws)
                Total-of-Dice = 0
REPEAT
    ADD 1 to Counter        (need to count how many grades are entered!)
    READ Dice-1 , READ Dice-2          (this is the input!)
    Total-of-Dice = Dice-1 + Dice-2
UNTIL  Total-of-Dice = 12        (if it is 12, it must be a double 6!, stop)
WRITE "The number of throws is " Counter        (this is the output!)
```

3.

Problem: You have 5 employees in your **Video shop**. Calculate the salary at the end of the month of each worker (they work 40 hours a week and earn 10 FL an hour). Also calculate the total of the money you have to pay at the end of the month.

Solution: Now you can see a solution with a FOR-loop. That's because we have to do something exactly 5 times! The FOR-loop has a built-in counter, so we don't have to manage that ourselves.

```
Initialise: Total-Salary = 0        (a variable, used for the total to pay)
Salary-one-Employee = 40 * 10 (calculate the salary of one worker!)
FOR  counter = 1 to 5       (here we make sure that the loop goes 5 times)
    ADD Salary-one-Employee to Total-Salary
END FOR                     (here we end the FOR-loop)
WRITE "The total to pay is " Total-Salary  (this is the output!)
```

4.

Problem: This is to practise the CASE statement: Accept the Number of the Month (from a keyboard). If the number = 1 then output January, if the number = 2 then . . .etc.

```
READ    Number-of-Month
CASE    Number-of-Month is :
    1 :    WRITE "It's January"
    2 :    WRITE "It's February"
    3 :    WRITE "It's March"
etc.
END CASE
```