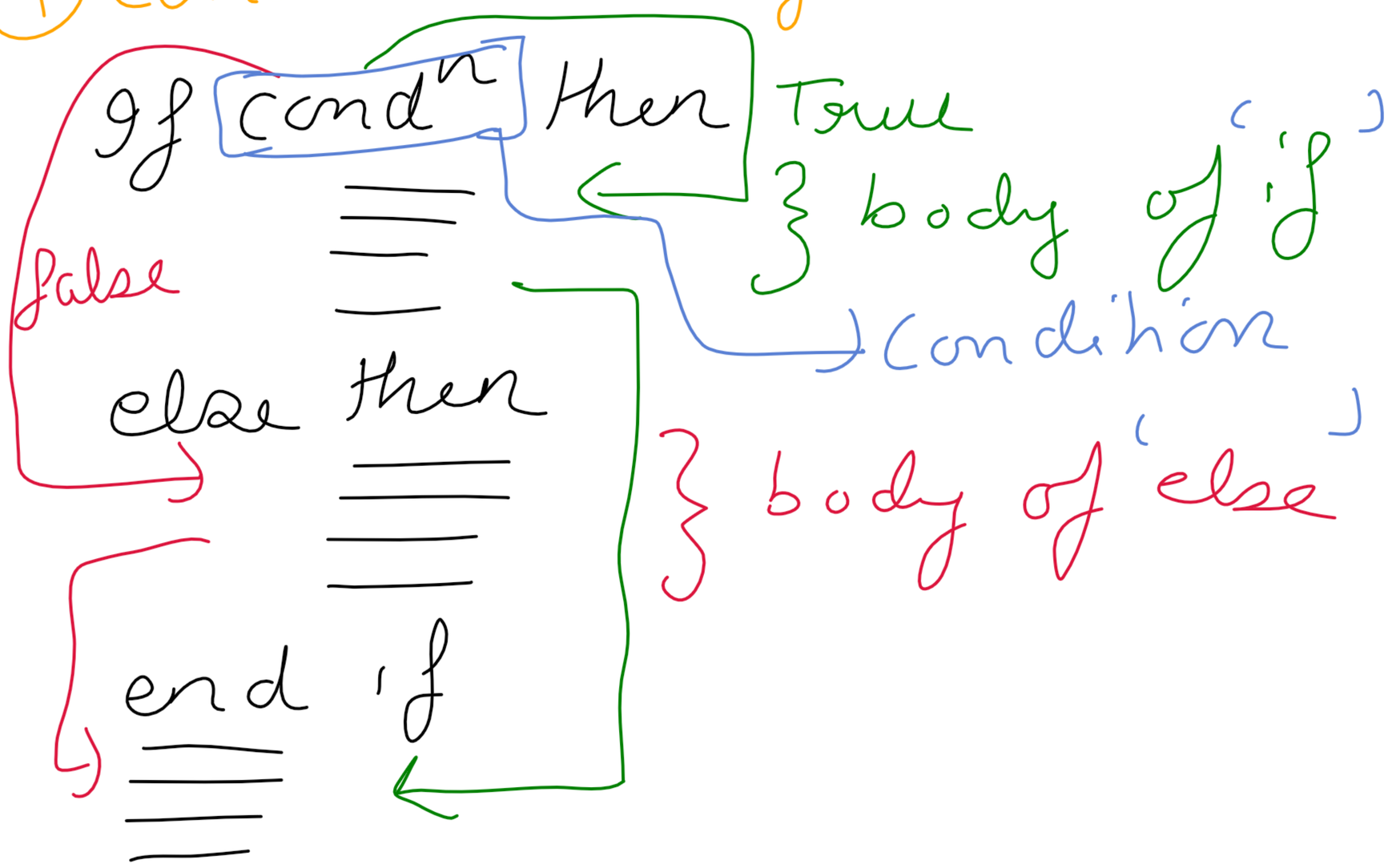
Introduction to Pseudo Code

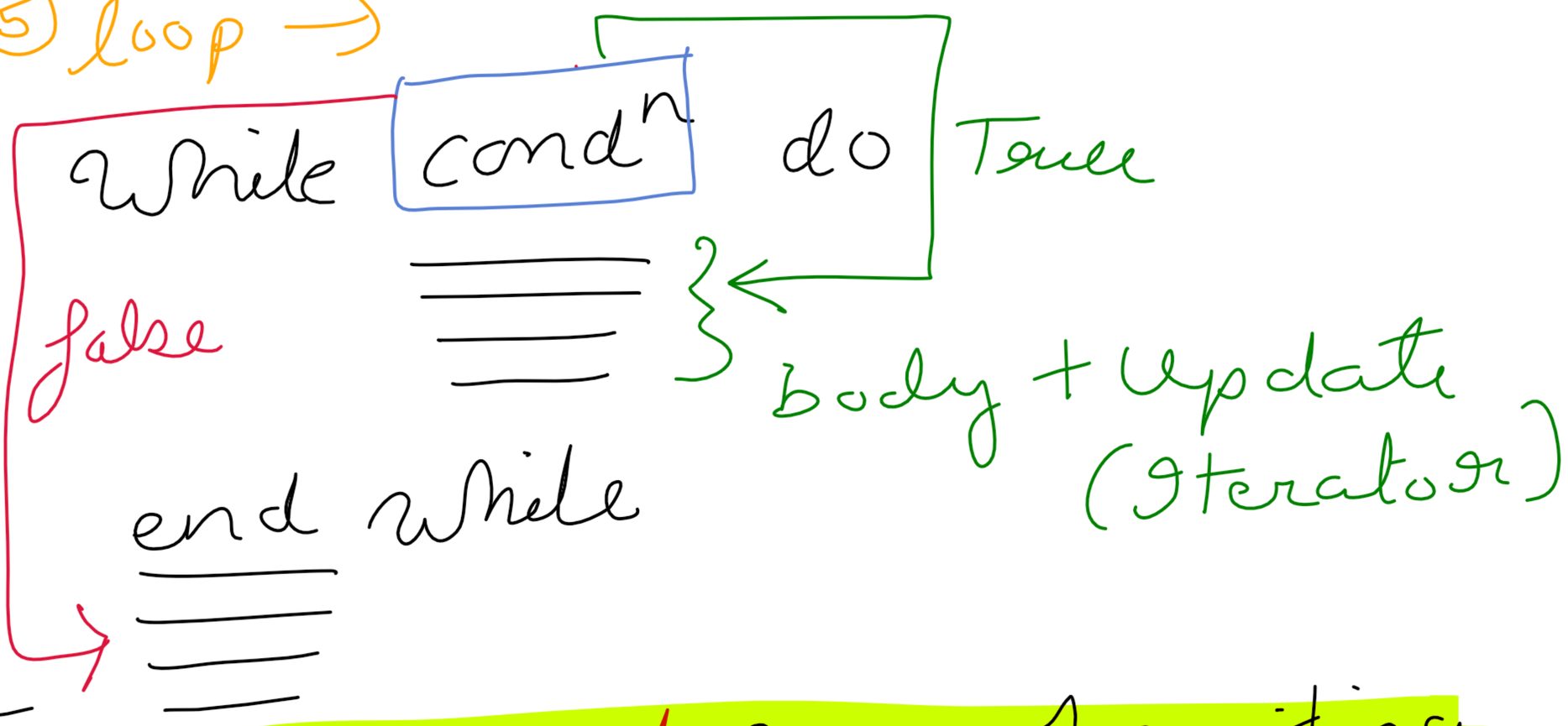
Pseudo Code :- It is a way of expressing an algo that involves usage plain language to describe the steps of an algorithm.

How to write Pseudo Code :-

1. Start / Terminate
   ↳ begin, end
2. Variable Assignment
   ↳ [var ← val] eg :- i ← 1
   j ← 0
3. Input / Output
   ↳ read var
   print var
4. Conditional logic ⇒



⑤ loop →

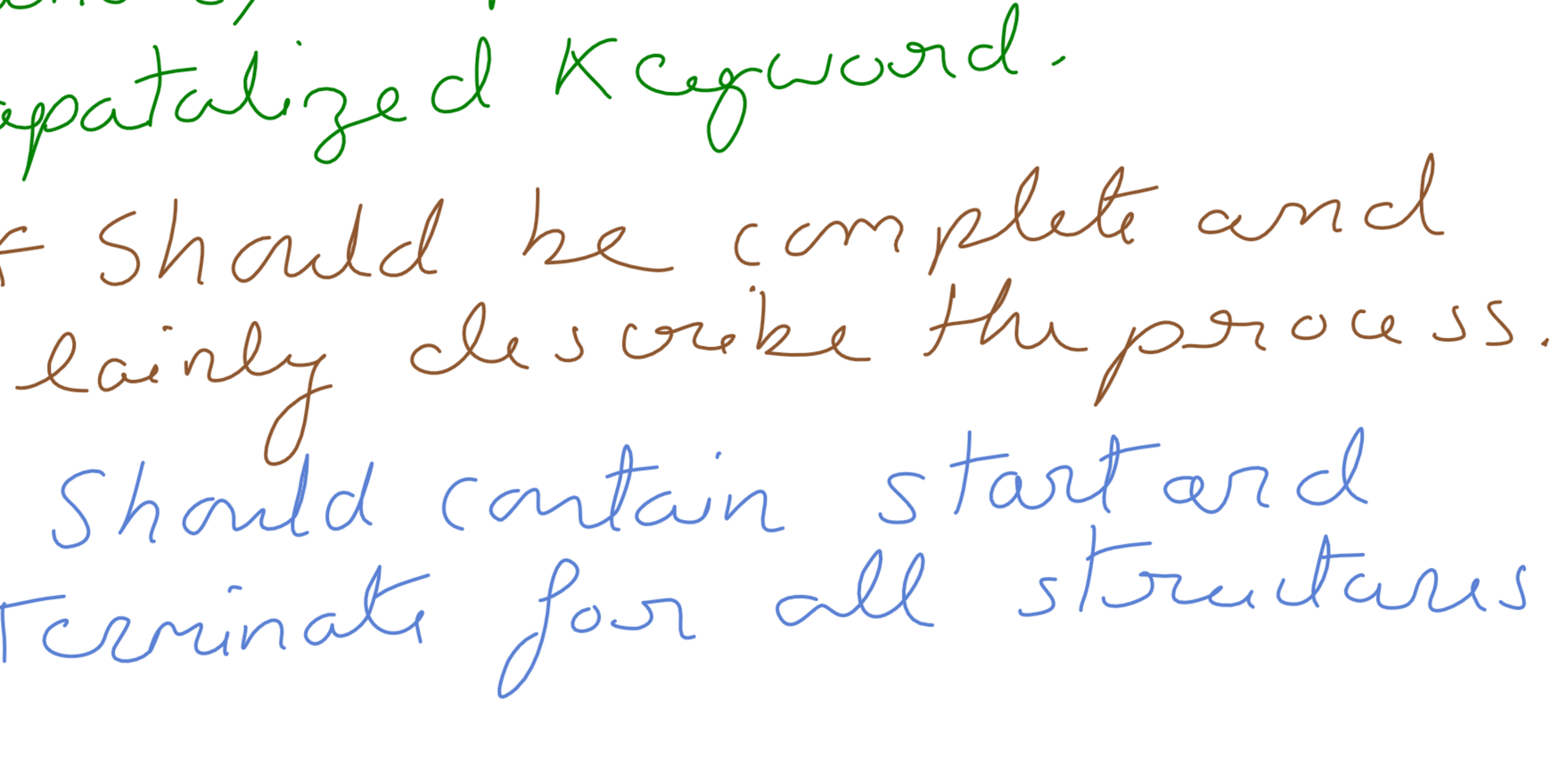


Good VS Bad Way of writing Pseudo code

✔ * IF/ELSE and loop structures should be used with clear indent-ation, simple names and, capatalized Keyword.

✔ * Should be complete and plainly describe the process.

✔ * Should contain start and Terminate for all structures.



Examples :-

① Largest of 'N' Numbers

Important

N = 5

Input :- [2, 4, 0, 8, 6]

output :- 8

code :-

Create lsf and check if lsf is (iterator)

```
BEGIN
read n
read x
lsf ← x
WHILE i<=n-1 do
    read x
    IF x > lsf then
    lsf ← x
    END IF
    i ← i+1
END WHILE
print(lsf)
END
```

Body of if

Body of while loop.

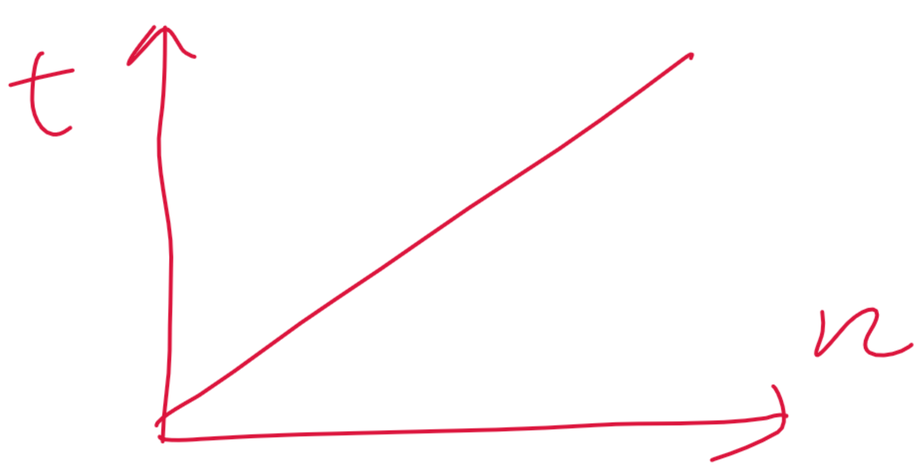
#iterations = n

Work done in each = const

∴ Total time :- n·c

~ O(n)

(linear time)



② Printing Stars (upto N)

Input :- N = 3 , Output :- * * *

Code :-

```
BEGIN
read n
j ← 1
WHILE j <= n do
    print "*"
    j ← j+1
END WHILE
END
```

To print N no. of stars while loop will run till j <= N

→ Condition

[Time ~ O(n)]

③ Matrix of stars (print N×N stars)

Input :- 3 , Output :-

```
* * *
* * *
* * *
```

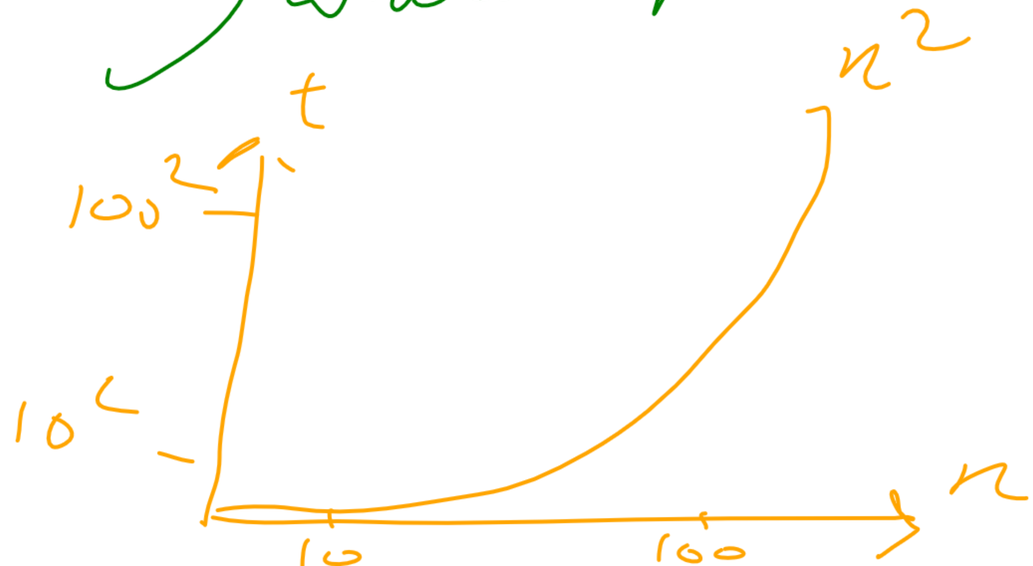
Code :-

```
BEGIN
read n
i ← 1
WHILE i <= n DO
    j ← 1
    WHILE j <= n DO
        print "*"
        j ← j+1
    END WHILE
    print 'In'
    i ← i+1
END WHILE
END
```

[Time : $O(n^2)$]

Body of Outer loop

Body of Inner while loop



'\n' VS endl

cout << '\n' is generally is more optimal because it only insert new line, while cout << endl insert new line and forces flush of output buffer, which can be slower.

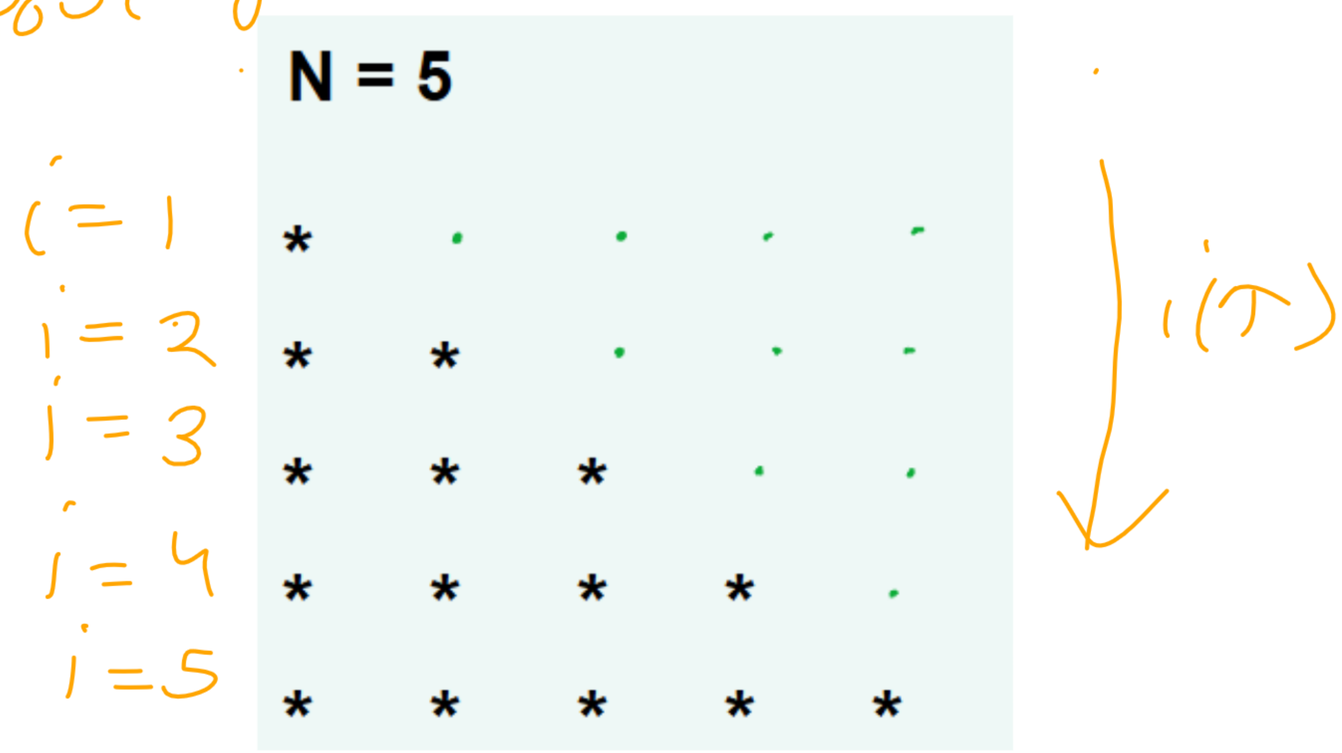
④ Right-Angled Triangle of Stars

Input :- N=4,

Thought Process :-

Output :-

for eg :-



$1 \leq i \leq n$

No of stars (i) ∝ i

Outer loop will run from 1→n and inner loop will run till (j <= i or)

Code :-

```
BEGIN
read n
i ← 1
WHILE (i <= n) DO
    j ← 1
    WHILE (j <= i) DO
        print '*'
        j ← j+1
    END WHILE
    print '\n'
    i ← i + 1
END WHILE
END
```

[Time = $O(n^2)$]

{ To print i stars for i^{th} row.

(5) Right Angled Triangle of stars - II

Input :- N = 4, Output :-

N = 5

```
        *
      * *
    * * *
  * * * *
* * * * *
```

i = 1

i = N

Thought Process

No of Spaces = n - i

No of stars = i

$[1 \leq i \leq n]$

Code:-

```
BEGIN
read n
i ← 1
WHILE i ≤ n DO
    j ← 1
    WHILE j <= n-i DO
        print ' '
        J ← J + 1
    END WHILE
    K ← 1
    WHILE K <= i DO
        print '*'
        K ← K + 1
    END WHILE
    print '\n'
    i ← i + 1
END WHILE
END
```

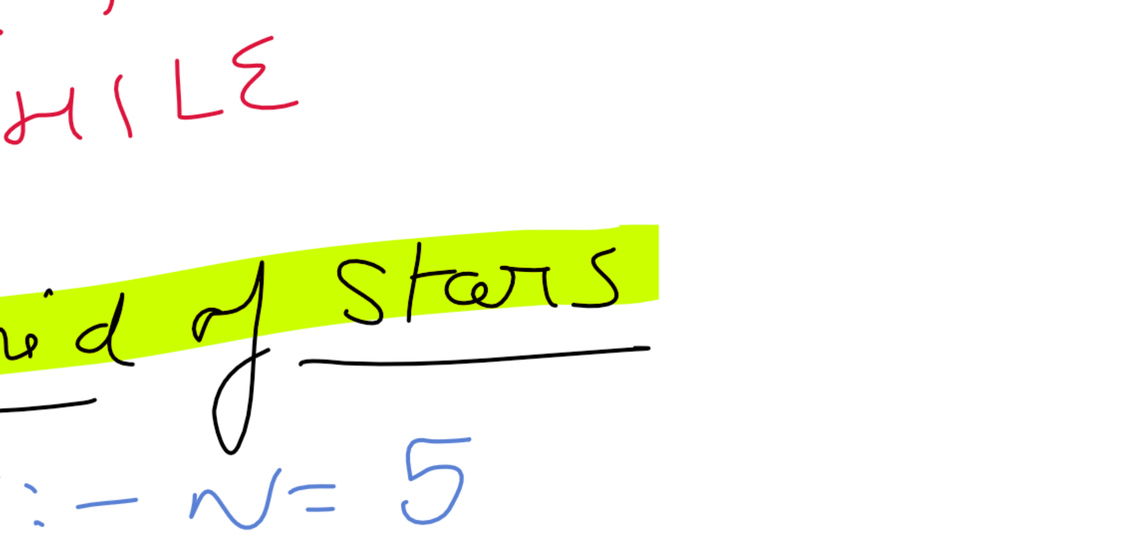
Print spaces

Print stars

⑥ Pyramid of Stars

Input :- N = 5

Output :-



We can, run three loop inside the outer loop as the pyramid will be broken down to three segments

Thought Process

In single iteration, we have to print,
(n-i) spaces, i (stars), i-1 (extra stars)

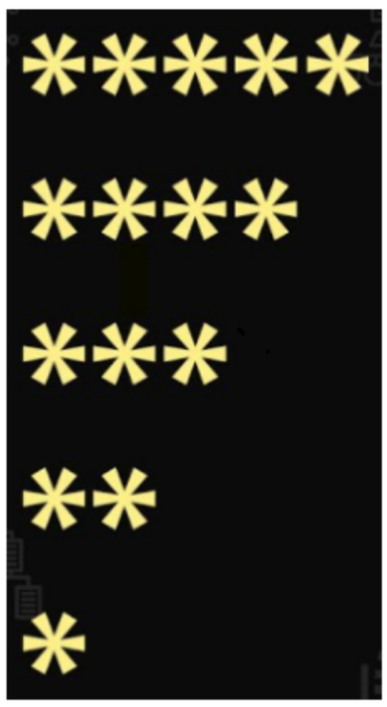
Code :-

```
BEGIN
read n
i ← 1
WHILE i <= n DO
    j ← 1
    WHILE j <= n-i DO
        print ' '
        j ← j+1
    END WHILE
    j ← 1
    WHILE j <= i DO
        print '*'
        j ← j+1
    END WHILE
    j ← 1
    WHILE j <= i-1 DO
        print '*'
        j ← j+1
    END WHILE
```

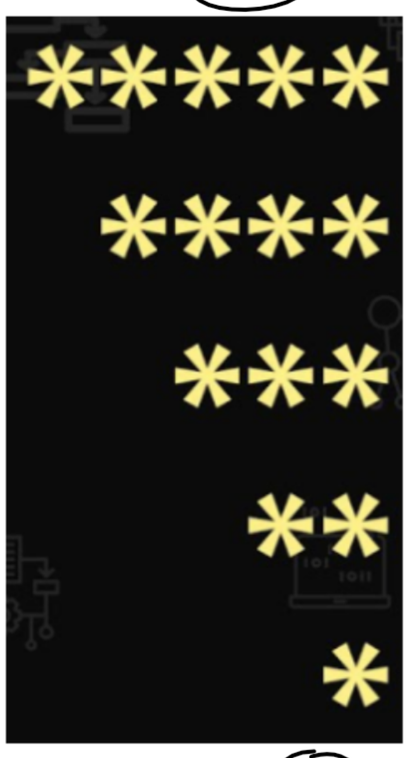
```
    i ← i+1
    print '\n'
END WHILE
END
```

HOME WORK

(1)



(2)



(3)

(4)

(5)

(6)

Q1:- (Inverted left aligned Triangle)

```
BEGIN
   READ  N // Number of rows / initial stars
  // Loop through each row, decreasing star count
   FOR row FROM N DOWNTO 1
   // Print stars for the current row
      FOR star_count FROM 1 TO row
         PRINT "*"
      END FOR
   PRINT NEWLINE // Move cursor to the next line
   END FOR
END
```

Q2. (Inverted Right Aligned Triangle)

```
BEGIN
   READ N // Get the number of rows/initial width

   // Outer loop: Controls the rows, counts down from
N to 1
   FOR i FROM N DOWNTO 1
      // Inner loop 1: Prints leading spaces
      FOR k FROM (N - i) TO 1
         PRINT " " // Print a single space
      END FOR

      // Inner loop 2: Prints the stars for the current row
      FOR j FROM i TO 1
         PRINT "*"
      END FOR

      // Move to the next line after printing spaces and
stars for the row
      PRINT NEWLINE
   END FOR
   END
```

Q3 (Print Pattern)

```
BEGIN
  DECLARE n AS INTEGER
  DECLARE x AS INTEGER

  SET x = 0
  READ n

FOR i FROM 0 TO n - 1 DO

 FOR j FROM 0 TO (n - i) - 1 DO
   PRINT "*"
 END FOR

 FOR h FROM 0 TO x - 1 DO
   PRINT " "
 END FOR

 FOR j FROM n + i TO (2 * n) - 1 DO
   PRINT "*"
```

```
    END FOR

    SET x = x + 2
    PRINT NEWLINE

  END FOR

END
```

Q4. Print Pattern

```
BEGIN
  DECLARE n AS INTEGER
  DECLARE x AS INTEGER

  SET x = (2 * (n - 1))
  READ n

  FOR i FROM 0 TO n - 1 DO

    FOR j FROM 0 TO i DO
      PRINT "*"
  END FOR

  FOR h FROM 0 TO x - 1 DO
    PRINT " "
  END FOR

  FOR j FROM 0 TO i DO
    PRINT "*"
  END FOR

  SET x = x - 2
    PRINT NEWLINE

  END FOR

END
```

Q5. (Parallelogram)

```
BEGIN
  DECLARE n AS INTEGER
  DECLARE space AS INTEGER

  READ n

  FOR i FROM 0 TO n - 1 DO

  FOR j FROM 0 TO i - 1 DO
    PRINT " "
  END FOR

  FOR k FROM 0 TO n - 1 DO
    PRINT "*"
  END FOR

  PRINT NEWLINE

END FOR
```

Q6. Inverse Parallelogram

```
BEGIN
  DECLARE n AS INTEGER
  READ n

  FOR i FROM 0 TO n - 1 DO

    FOR j FROM 0 TO (n - i - 2) DO
      PRINT " "
    END FOR

    FOR k FROM 0 TO n - 1 DO
      PRINT "*"
    END FOR

    PRINT NEWLINE

  END FOR

END
```