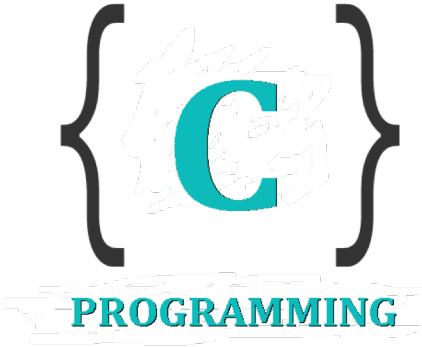



# IT 105: Introduction to Programming

Dr. Manish Khare
Dr. Saurabh Tiwari

Lecture 2

# Last Lecture's Exercise

Write an algorithm to compute the quotient and reminder when a given integer x (x >=0) is divided by another integer y (y>0)

Assume that only addition and subtraction are available as primitive arithmetic operators.

# Last Lecture's Exercise

**Properties of Quotient and Reminder:**

If q and r are quotient and reminder respectively obtained after dividing x by y:

- $q * y + r = x$
- $r < y$

# Solution

1. Read the values of x and y
2. Set value of q to 0.
3. While (x >= y) do (steps 3.1 and 3.2)
    1. x ← x – y;
    2. q ← q + 1
4. Print results: reminder is x and quotient is q.

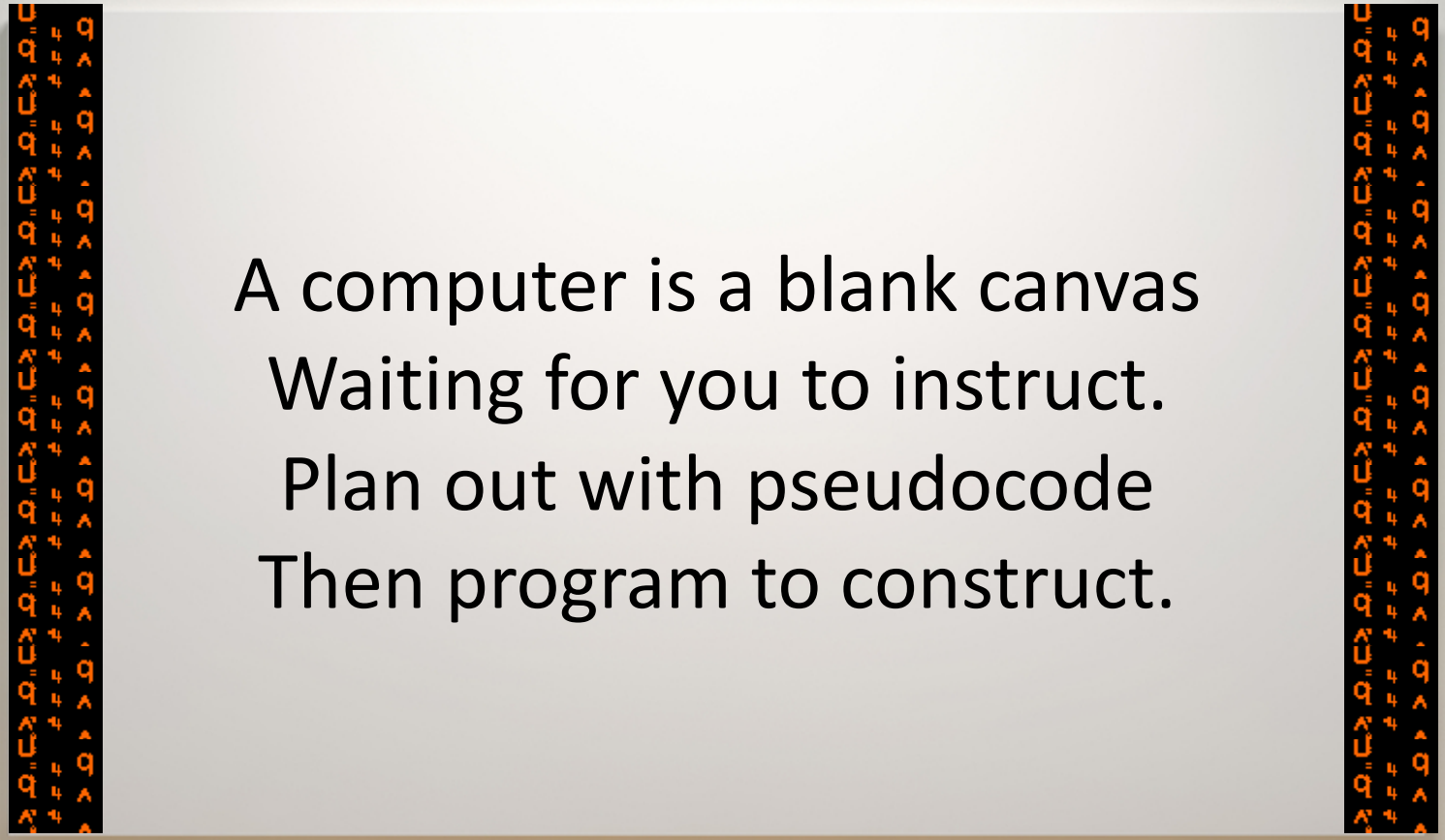
# Some Terminologies

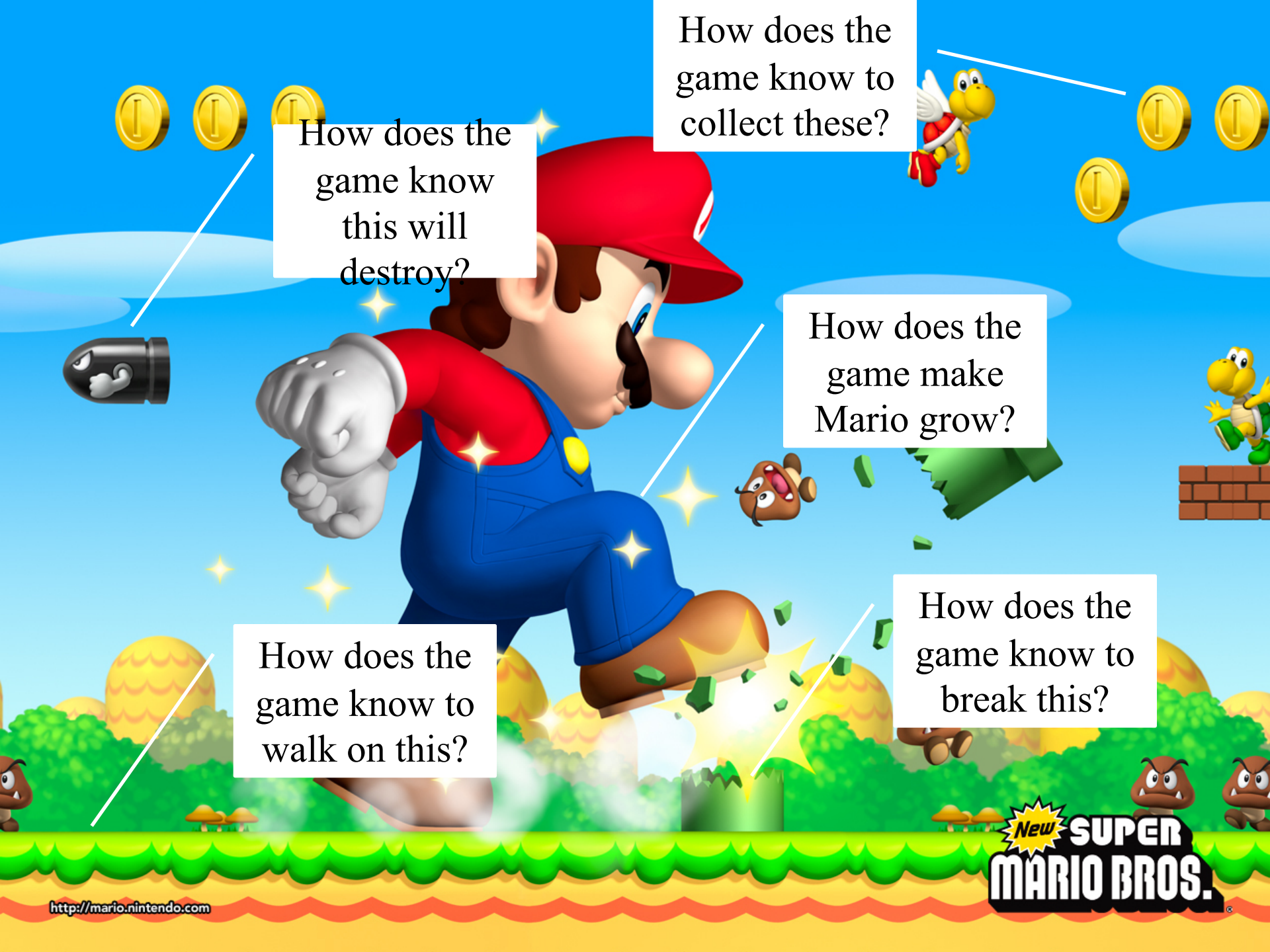
- Algorithm / Flowchart
  - A step-by-step procedure for solving a particular problem.
  - Should be independent of the programming language.

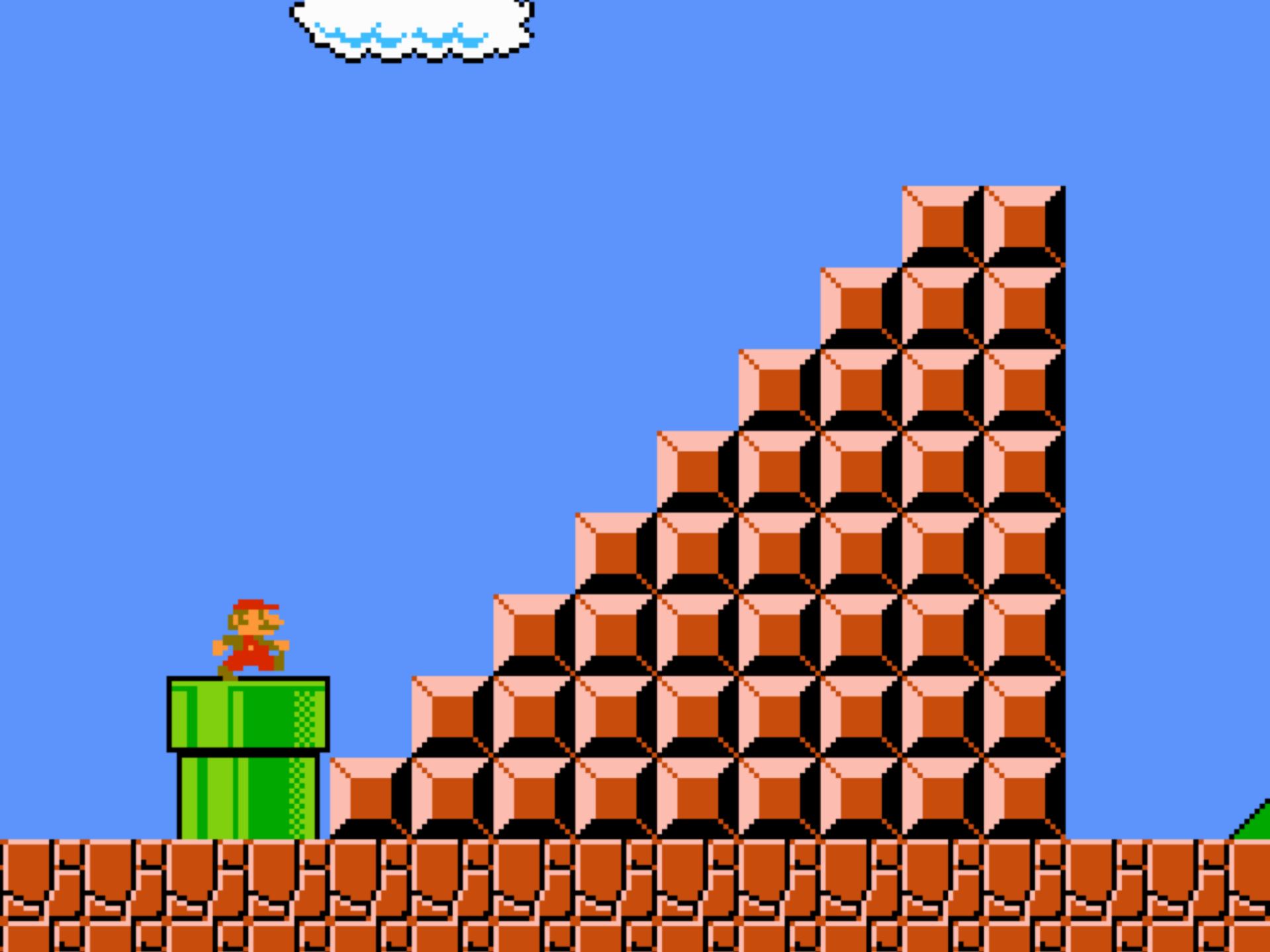
- Program
  - A translation of the algorithm/flowchart into a form that can be processed by a computer.
  - Typically written in a high-level language like C, C++, Java, etc.

# Problem Solving

- Step 1:
  - Clearly specify the problem to be solved.
- Step 2:
  - Draw flowchart or write algorithm.
- Step 3:
  - Convert flowchart (algorithm) into program code.
- Step 4:
  - Compile the program into object code.
- Step 5:
  - Execute the program.


A computer is a blank canvas
Waiting for you to instruct.
Plan out with pseudocode
Then program to construct.


How does the game know to collect these?
How does the game know this will destroy?
How does the game make Mario grow?
How does the game know to walk on this?
How does the game know to break this?
New SUPER MARIO BROS.
http://mario.nintendo.com



# Tower of Hanoi

# How do I get Minion Stuart to move to D1?

| | A | B | C | D |
|---|---|---|---|---|
| 1 |  | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

Move right 3 squares

# How do I get Minion Stuart to move to D1 then to D4?

| | A | B | C | D |
|---|---|---|---|---|
| 1 |  | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

Move right 3 squares
Move down 3 squares

# How do I get Minion Stuart to move to D1 then to D4, then move to A4 and finally to A1?

| | A | B | C | D |
|---|---|---|---|---|
| 1 |  | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |

Move right 3 squares
Move down 3 squares
Move left 3 squares
Move up 3 squares

# What have we just done

- You have created a series of instructions to solve a given problem
- This is called an **Algorithm**
- When we write it in a list of instructions it is called **Pseudocode**
- Computer Programmers use pseudocode to help plan out the code they will need.
  - For game making
  - Creating websites
  - Control software – robots / machinery
  - ANYTHING where planning is needed = pseudocode is used to layout the tasks/actions

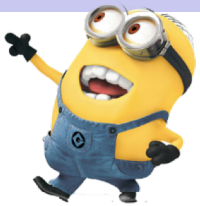
In your workbooks I want you to think about how to get Minion Stuart to move around the squares and collect the bananas. You will need to move him and also add actions needed to pick up the items.

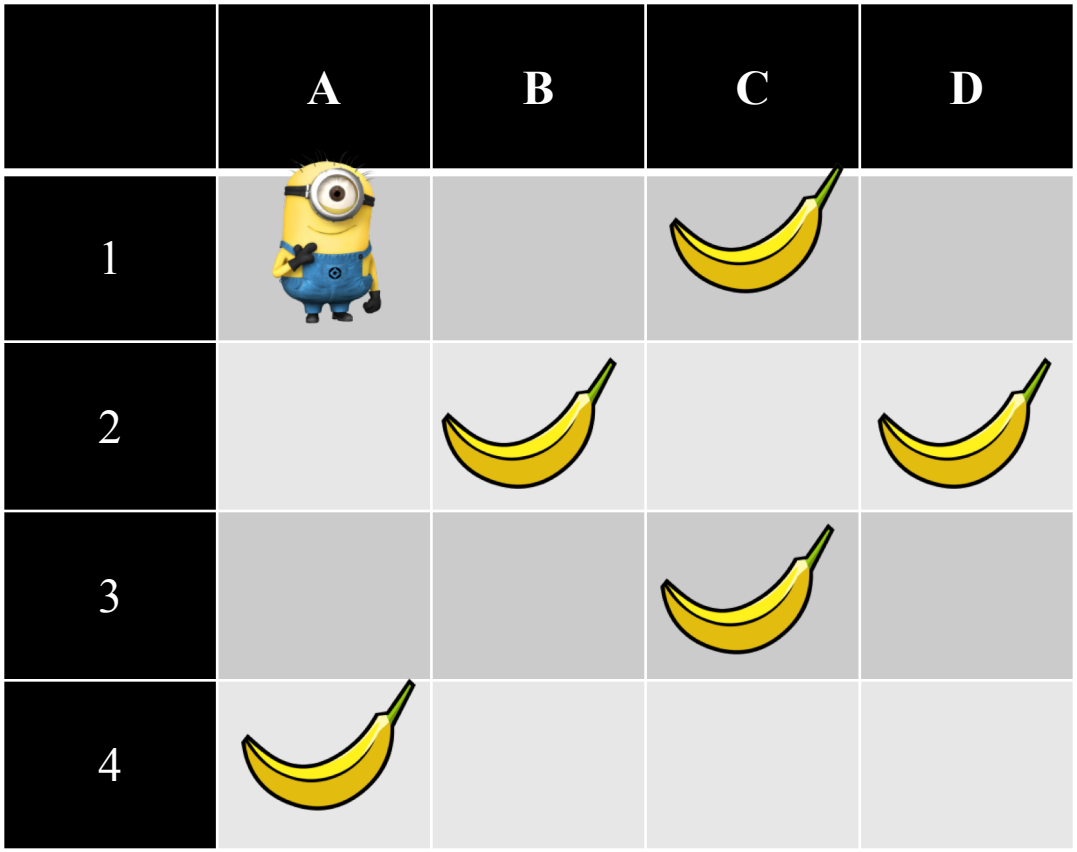
| | A | B | C | D |
|---|---|---|---|---|
| 1 |  | |  | |
| 2 | |  | |  |
| 3 | | |  | |
| 4 |  | | | |

## Your Turn

Extension = Can you add a 'question to say if the minion reaches a banana?

# Algorithm Solution written in pseudocode






1. Move right 2 squares
2. Pick up banana
3. Move right 1 square
4. Move down 1 square
5. Pick up banana
6. Move left 2 squares
7. Pick up banana
8. Move down 1 square
9. Move right 1 square
10. Pick up banana
11. Move down 1 square
12. Move left 2 squares
13. Pick up banana

# Can you ask a question?

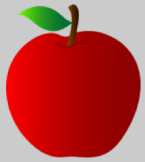
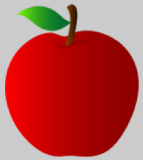
- When we think about the way Minion Stuart moves across could we ask a question as he moves from one side to the other?

1. Move across 3 squares
2. If you reach a banana
   a. Pick it up

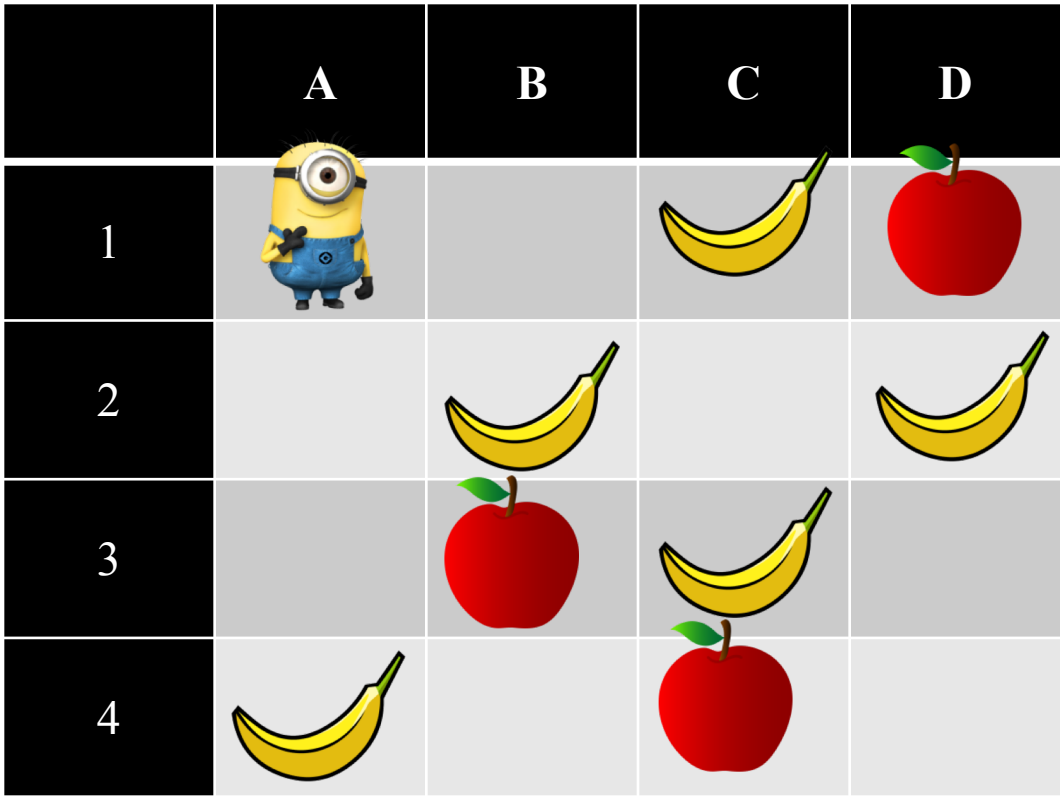
Ask the question – try and start it with an 'IF'

Ask the question – try and start it with an 'IF'

# Can we ask a question?

| | A | B | C | D |
|---|---|---|---|---|
| 1 |  | |  |  |
| 2 | |  | |  |
| 3 | |  |  | |
| 4 |  | |  | |

# Possible algorithm in pseudocode




1. Move right 3 squares
2. IF Minion reaches a banana
   a. THEN pick it up
3. IF Minion reaches an apple
   a. THEN leave it

This would continue to cover the whole board

This form of pseudocode would help a game designer plan out the code they would need to write to create it.

# Pseudocode: Keywords

- **begin ------ end** : begin is the first statement and end is the last statement of the pseudocode. All the instructions used in pseudocode is to be written between begin----end block.
- **Read:** Reading two values from the input given by the user
- **Compute :** The keyword compute is used for calculation of the result of the given expression.
- **Print :** It is used to display the output of the program
- **if--- else--- endif :** It is a decision construct used different action has to be taken depending on the condition.
- **while---- do :** This is also an iterative construct similar to do---while construct. The only difference is that in this construct, the testing statement is present at the top of set of the iterative statements.

# Pseudocode: Example

Write the pseudocode to accept two numbers and displays their sum and difference.

Begin *//starting the pseudocode*
Read first_number, second_number *//input two numbers*
Compute sum as first_number + second_number *// process of adding the given two numbers*
Compute difference as first_number - second_number *//process of calculating difference of the given two numbers*
Print 'The sum of the given two numbers is' sum *//Output to display the sum*
Print 'The difference of the given two numbers is' difference *//Output to display the difference*
End *//End of the pseudocode*

# Decision Structures

- The expression A>B is a logical expression

*it describes a **condition** we want to test*

- ***if A>B is true (if A is greater than B)** we take the action on left*

print the value of A

- ***if A>B is false (if A is not greater than B)** we take the action on right*

print the value of B

# if-then-else Structure

- The structure is as follows

***If condition then***

***true alternative***

***else***

***false alternative***

***endif***

| Relational Operators | |
|---|---|
| **Operator** | **Description** |
| > | **Greater than** |
| < | **Less than** |
| = | **Equal to** |
| ≥ | **Greater than or equal to** |
| ≤ | **Less than or equal to** |
| ≠ | **Not equal to** |

# Nested if Structure

- One of the alternatives within an IF–THEN–ELSE statement
  - may involve further IF–THEN–ELSE statement

# Loop Structure

- For Loop
- While… Do Loop
- Do… While Loop

# Exercise

- pseudocode for **compute the area of a rectangle**

*Begin*

*Get the length, l, and width, w*

*Compute the area = l*w*

*Display the area*

*End*

- pseudocode for **compute the perimeter of a rectangle**:

*Begin*

*Enter length, l*

*Enter width, w*

*Compute Perimeter = 2*l + 2*w*

*Display Perimeter of a rectangle*

*end*

- pseudocode for Calculate Pay – sequence

```
Begin
input hours
input rate
pay = hours * rate
print pay
End
```

## pseudocode for Calculate Pay with Overtime – selection

```
Begin
input hours, rate
if hours ≤ 40 then
      pay = hours * rate
else
      pay = 40 * rate + (hours – 40) * rate * 1.5
print pay
End
```

# Within Class Exercise

- Larger of two numbers
- *Roots of a quadratic equation*
- Grade Computation

$MARKS \geq 90$ → Ex

$89 \geq MARKS \geq 80$ → A

$79 \geq MARKS \geq 70$ → B

$69 \geq MARKS \geq 60$ → C

$59 \geq MARKS \geq 50$ → D

$49 \geq MARKS \geq 35$ → P

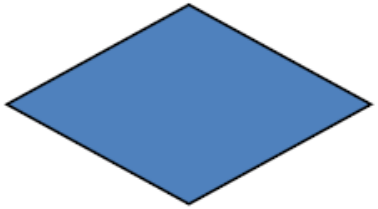
$34 \geq MARKS$ → F

# Flow Chart

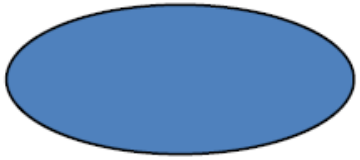
# Flow Chart: Basic Symbol

Computation

Input / Output



Decision Box

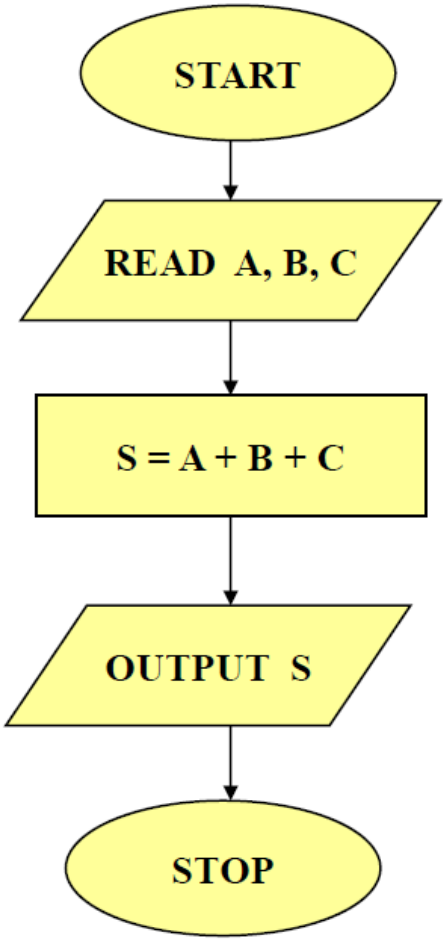


Start / Stop

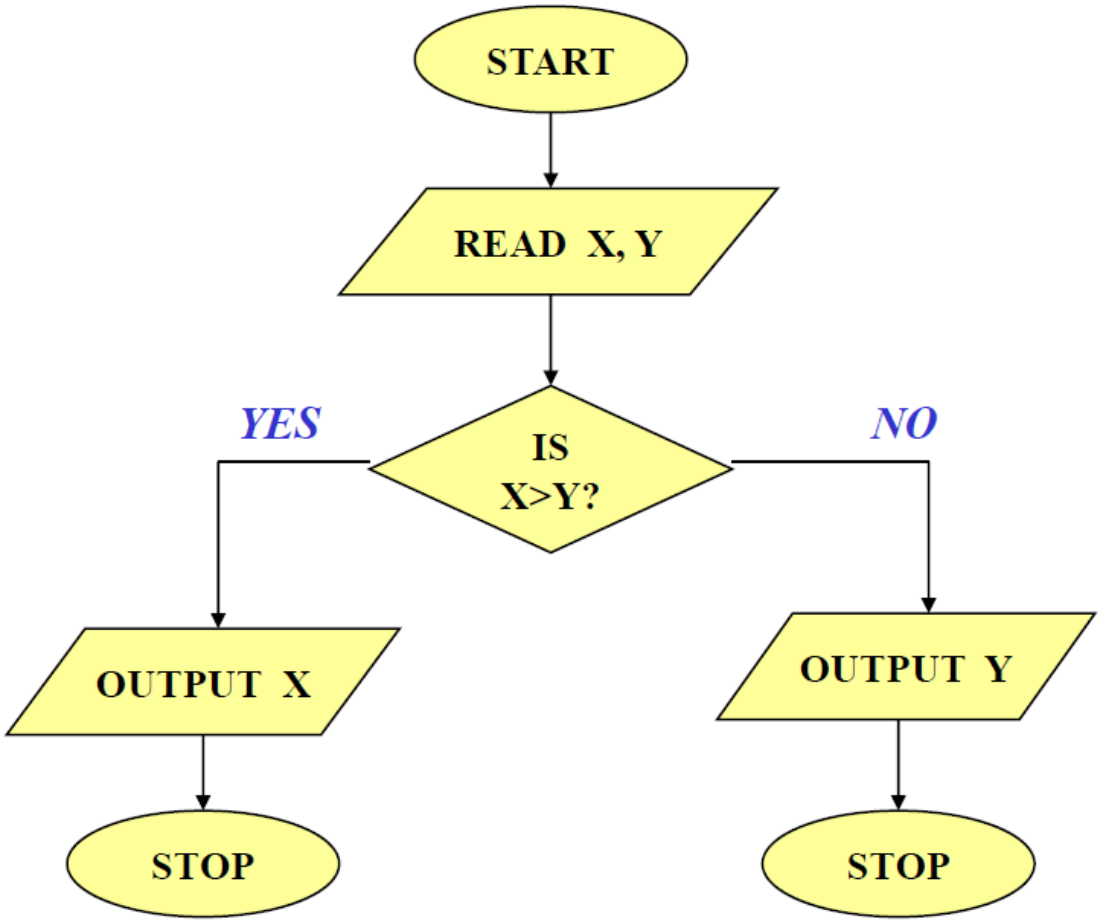
Flow of control

Connector

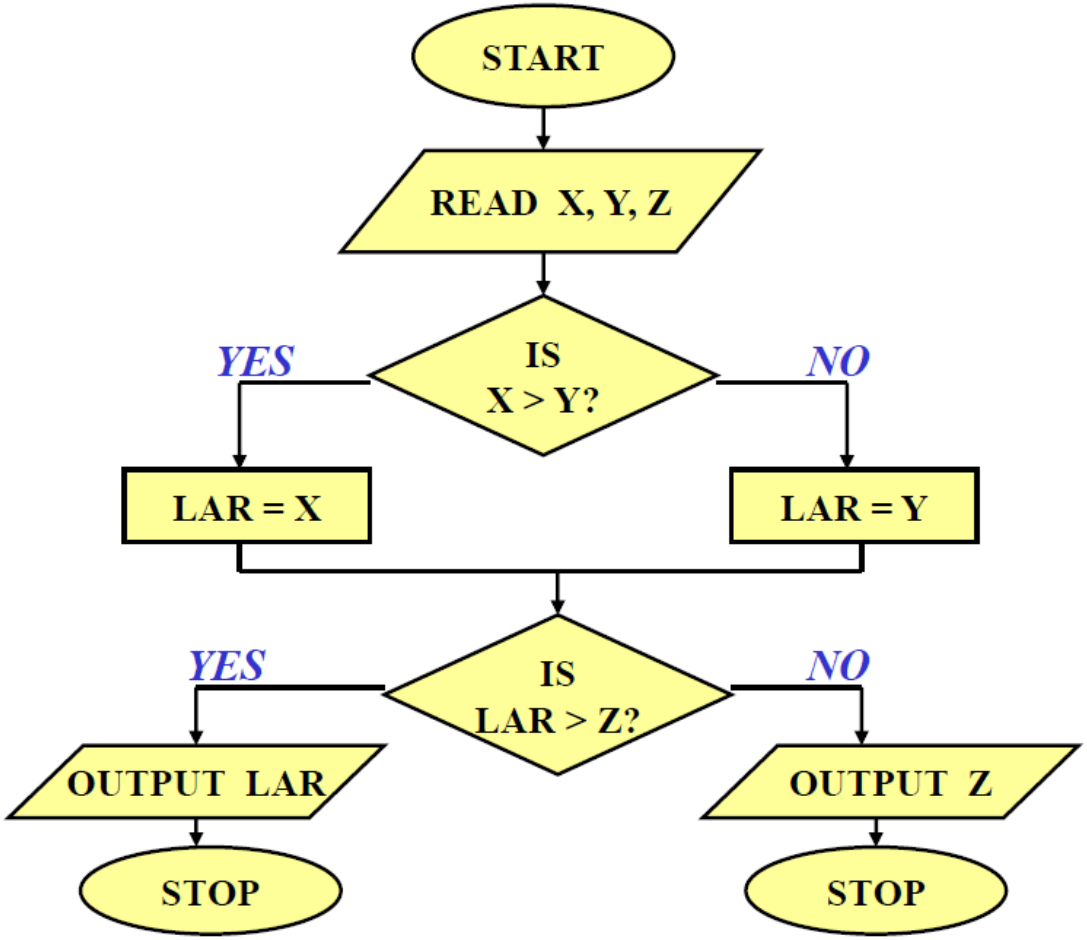
## **Example:** Adding Three Numbers



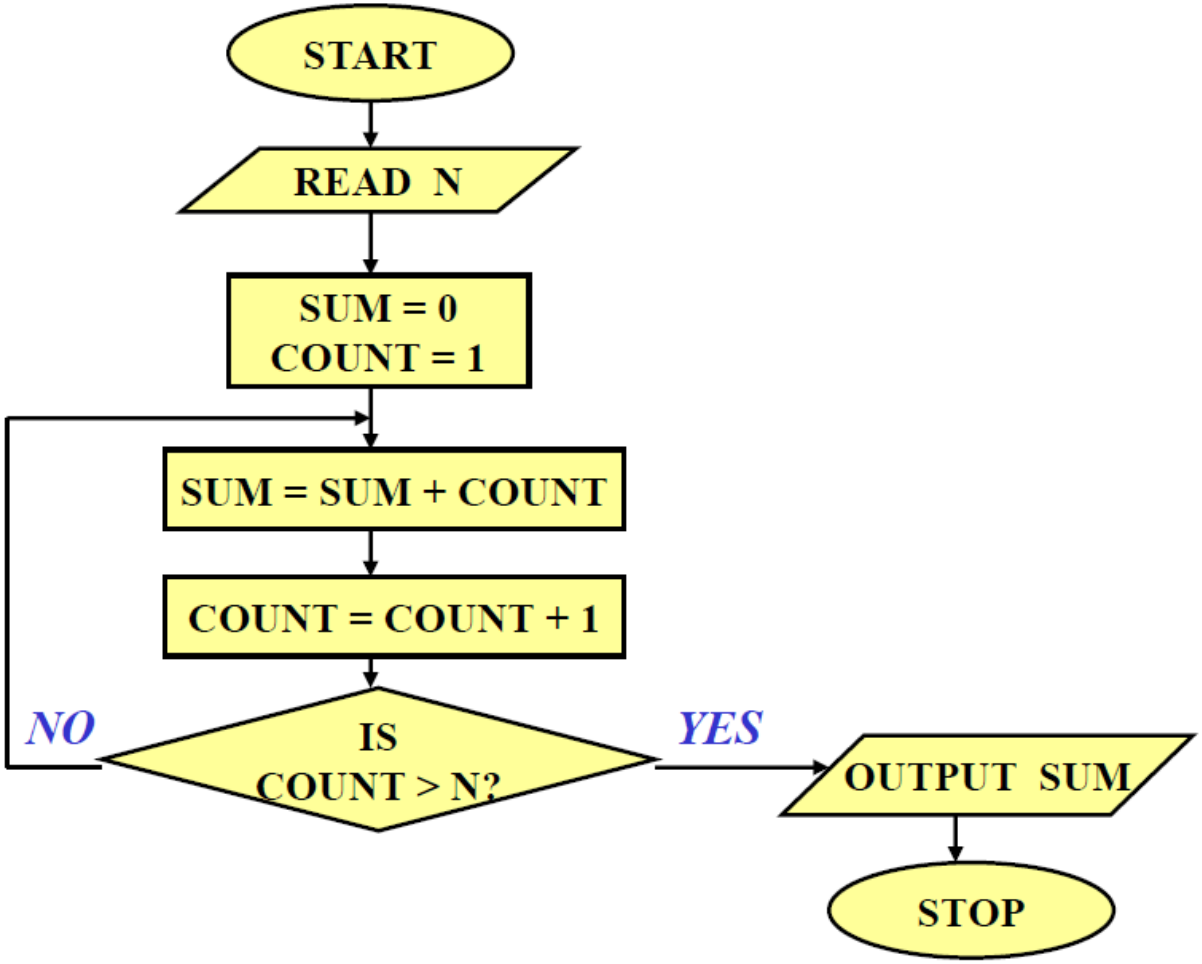
## Example: Larger of two numbers



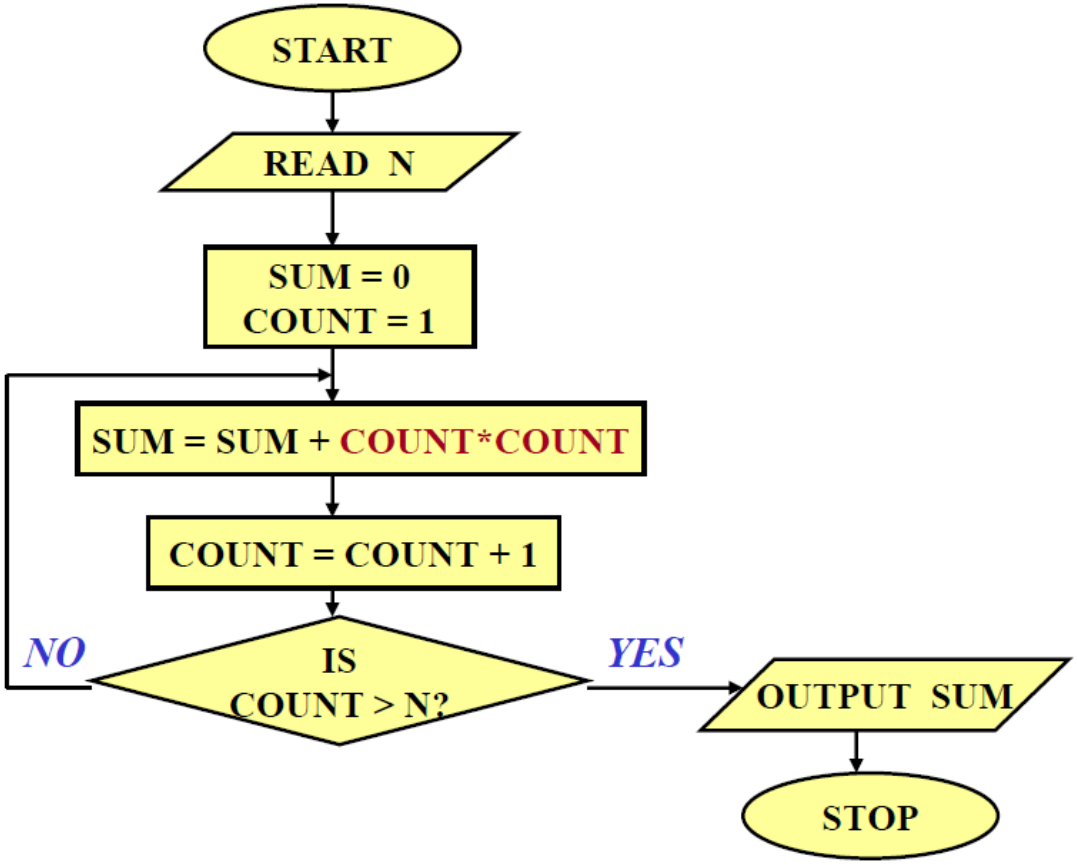
## Example: Largest of three numbers



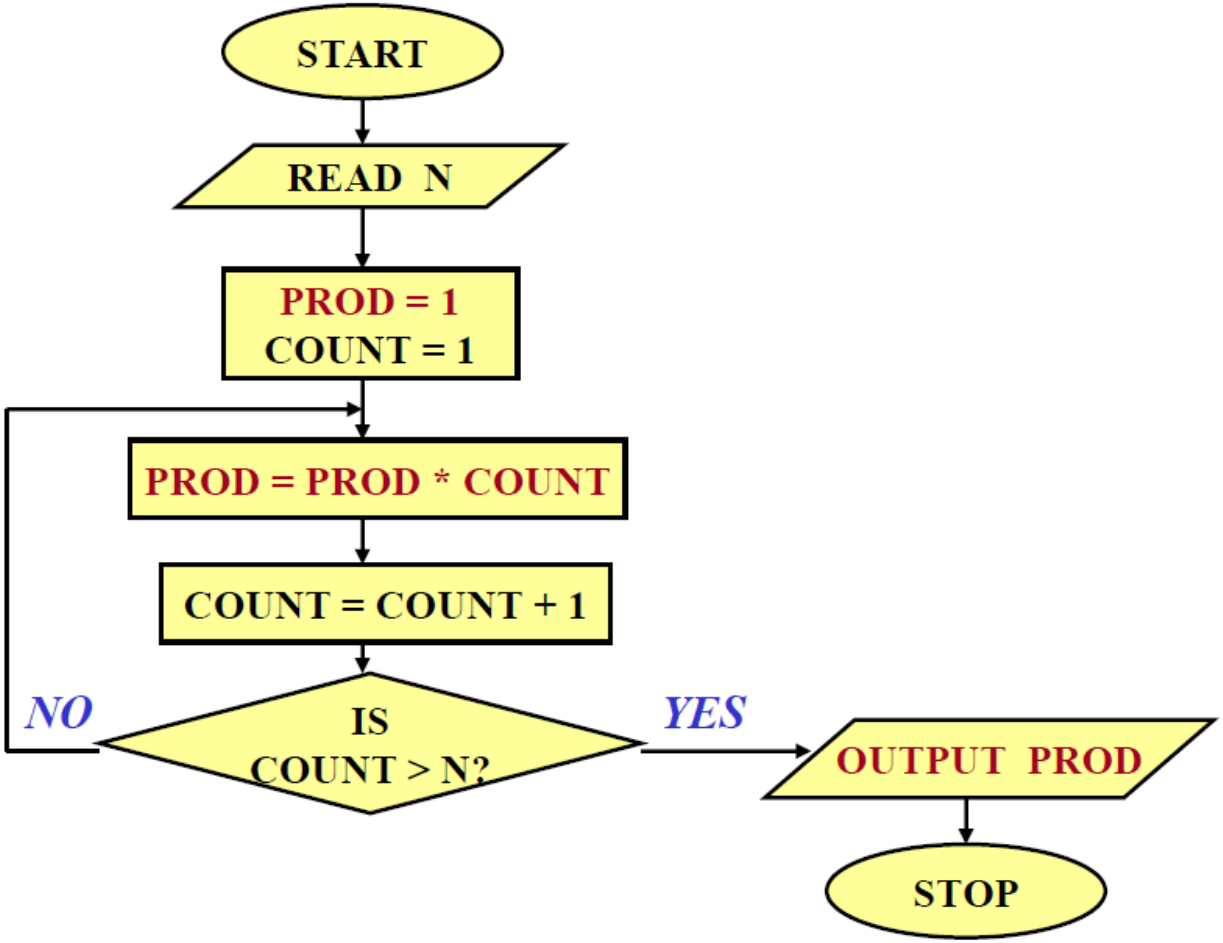
**Example:** Sum of first N natural numbers



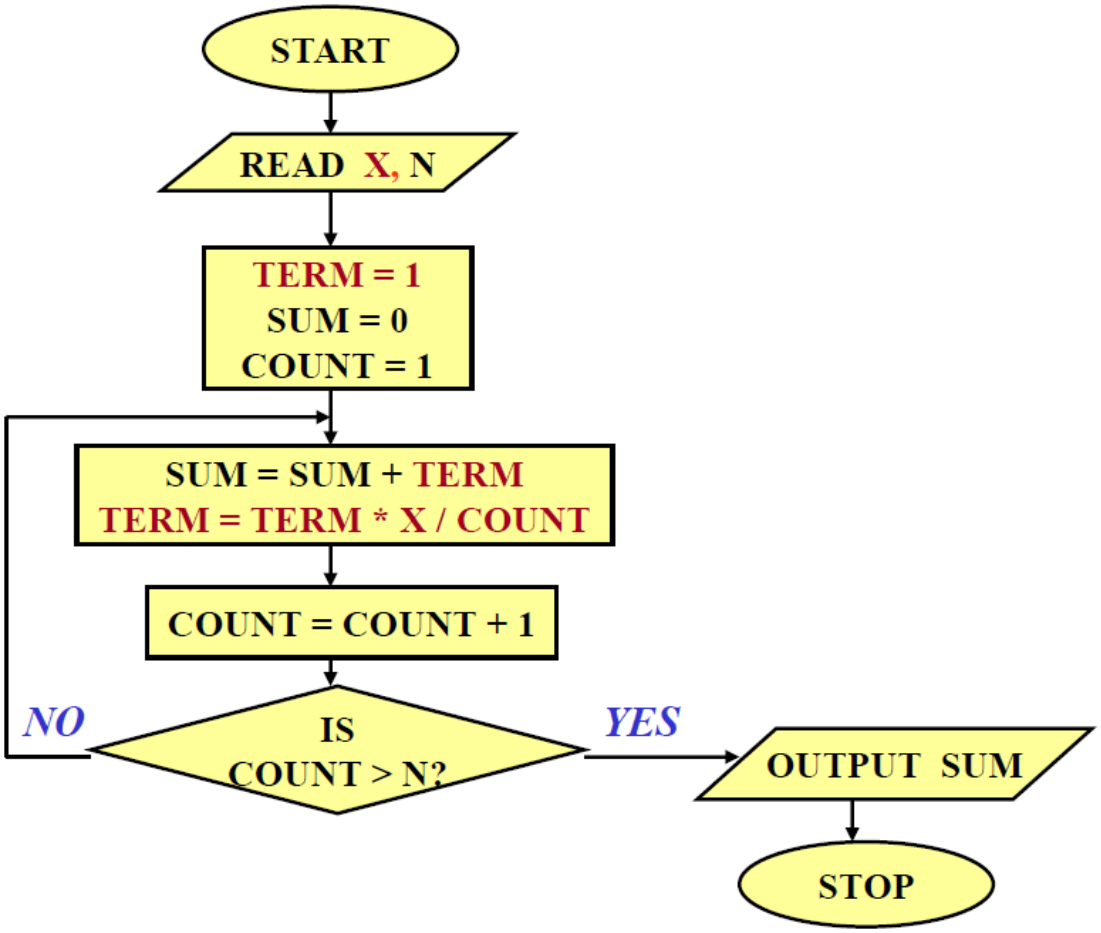
Example: $SUM = 1^2 + 2^2 + 3^2 + N^2$



## **Example:** Computing factorial

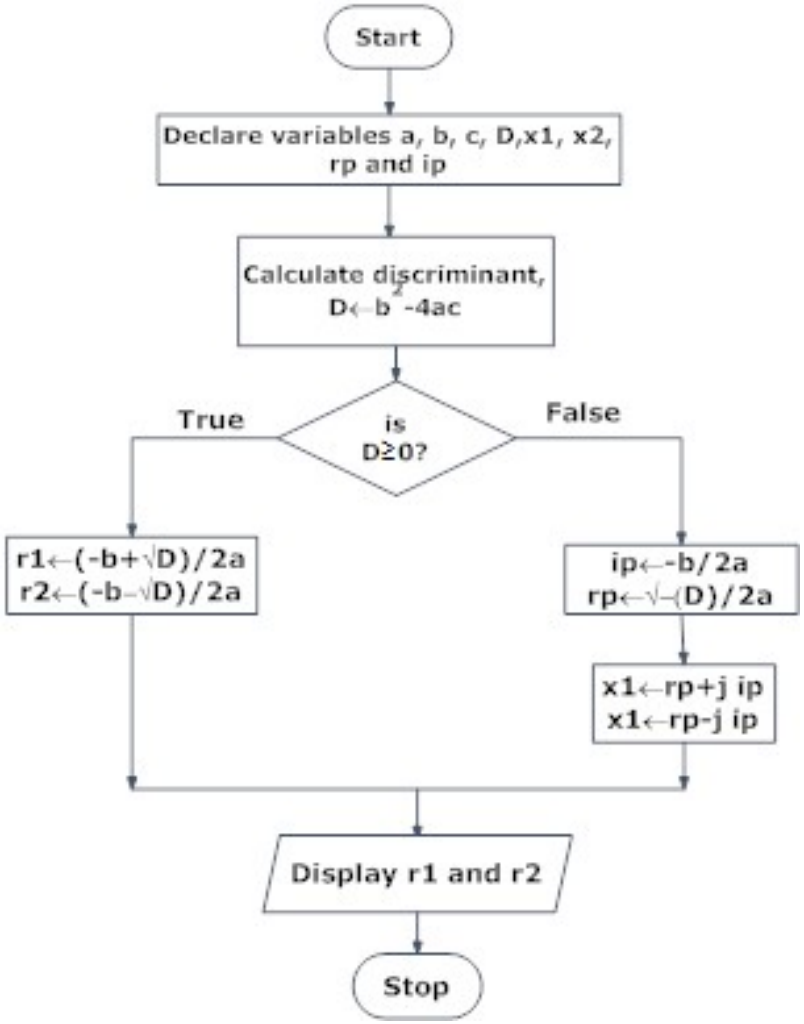


## Example: Computing $e^x$ series up to N terms



Example: *Roots of a quadratic equation*

$$ax^2 + bx + c = 0$$



**Example:** Grade Computation

$MARKS \geq 90$ → Ex

$89 \geq MARKS \geq 80$ → A

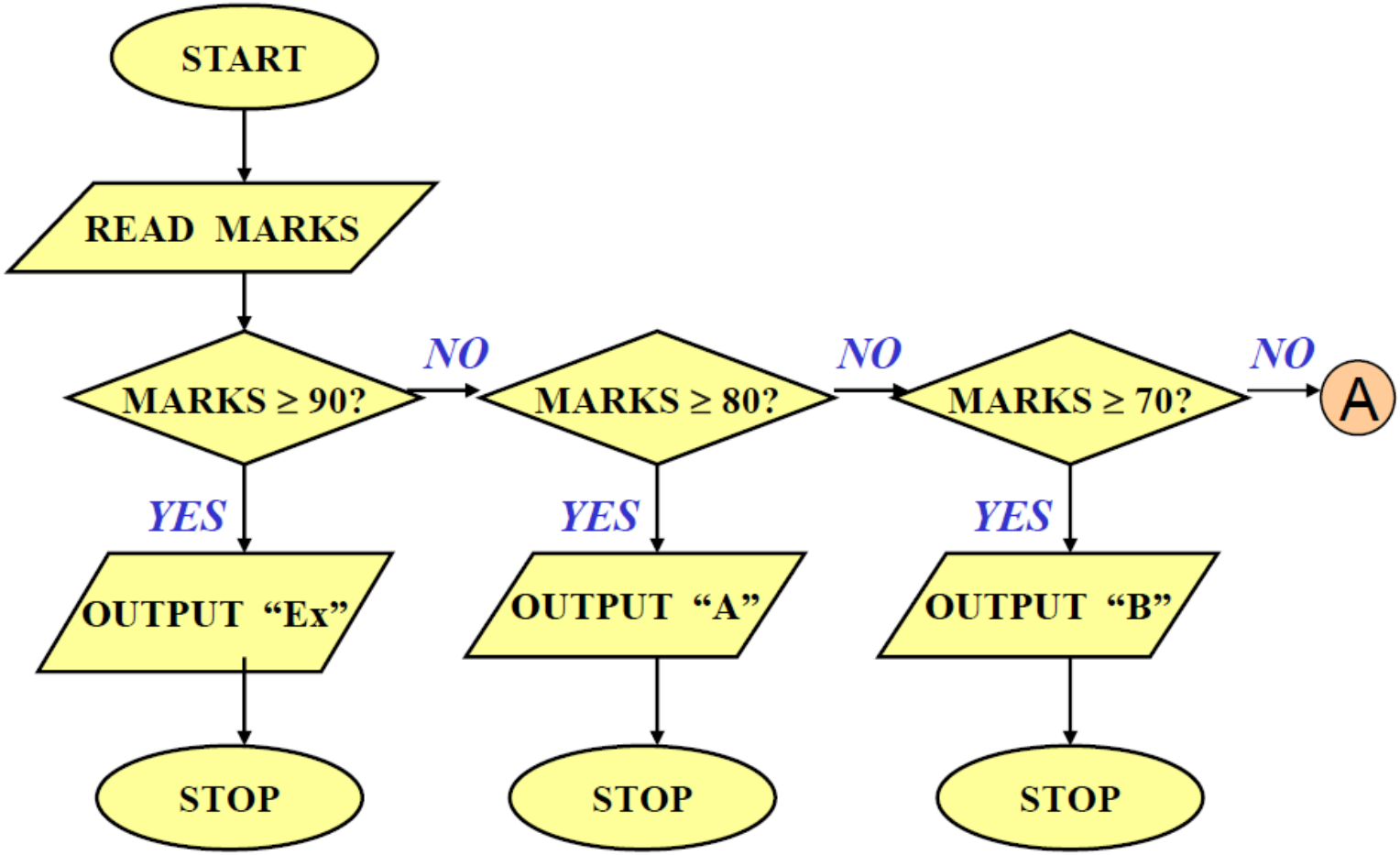
$79 \geq MARKS \geq 70$ → B

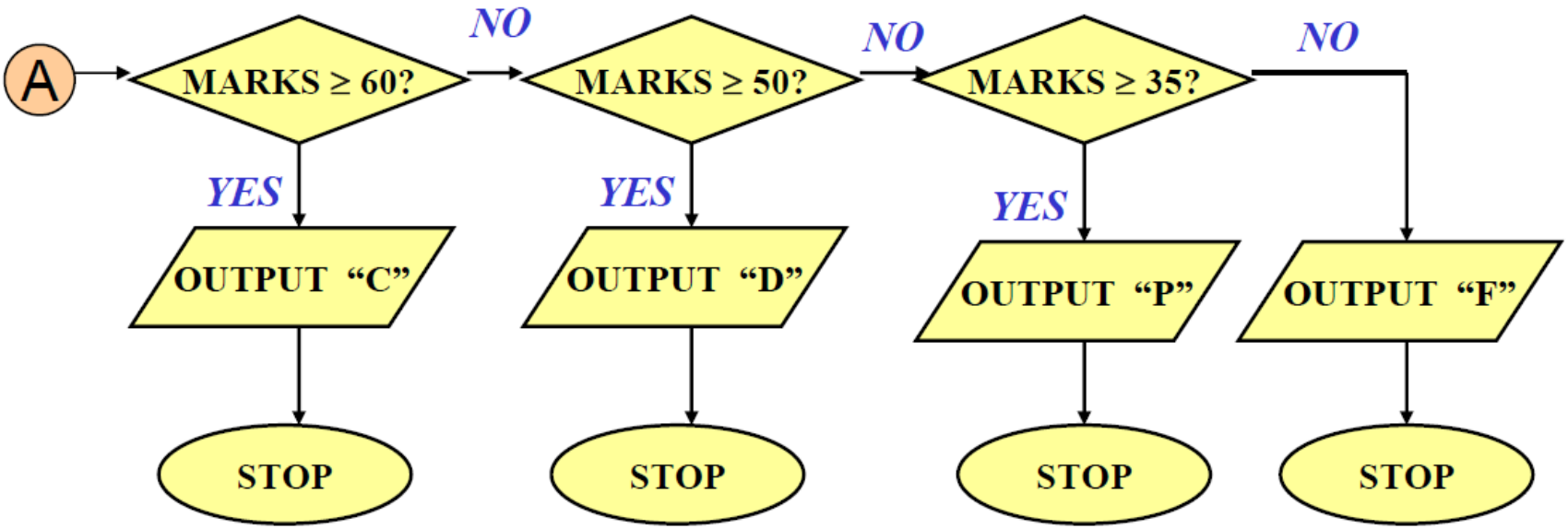
$69 \geq MARKS \geq 60$ → C

$59 \geq MARKS \geq 50$ → D

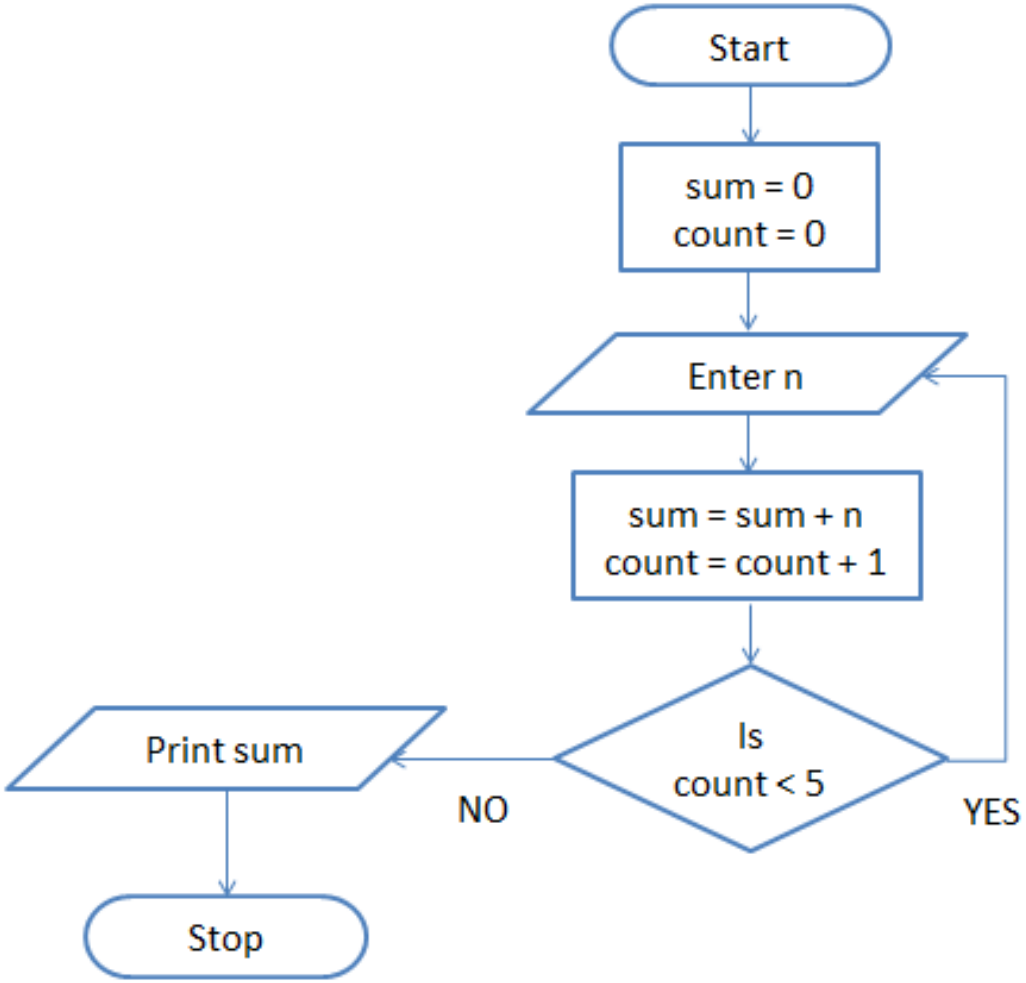
$49 \geq MARKS \geq 35$ → P

$34 \geq MARKS$ → F

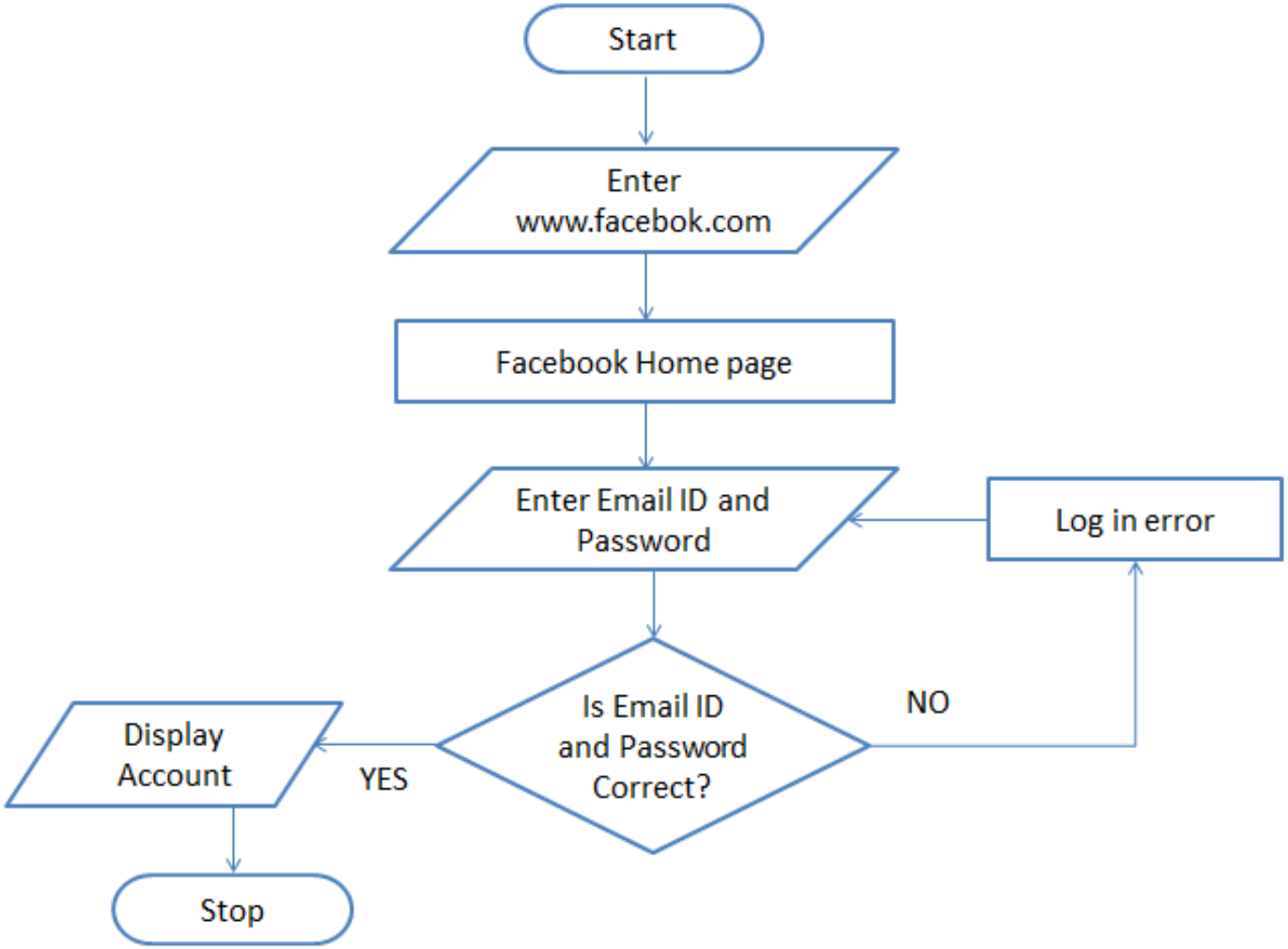

START
READ MARKS
MARKS ≥ 90?
NO
MARKS ≥ 80?
NO
MARKS ≥ 70?
NO
A
YES
OUTPUT "Ex"
STOP
YES
OUTPUT "A"
STOP
YES
OUTPUT "B"
STOP


A
MARKS ≥ 60?
NO
MARKS ≥ 50?
NO
MARKS ≥ 35?
NO
YES
OUTPUT "C"
YES
OUTPUT "D"
YES
OUTPUT "P"
OUTPUT "F"
STOP
STOP
STOP
STOP

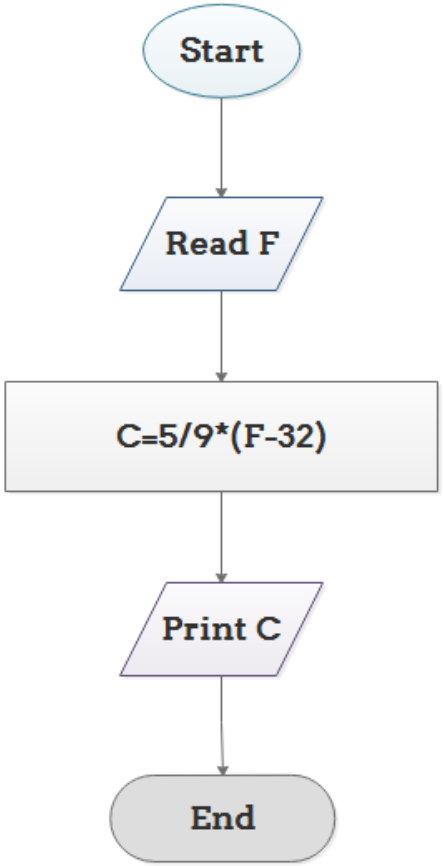
**Example:** Find the sum of 5 numbers



**Example:** Draw a flowchart to log in to facebook account

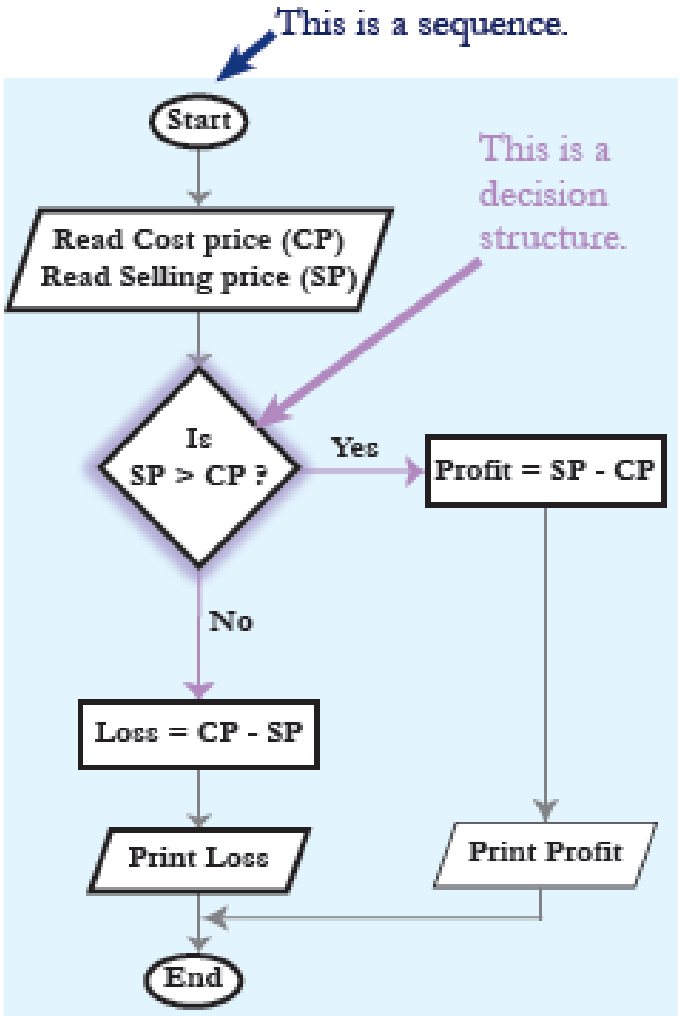


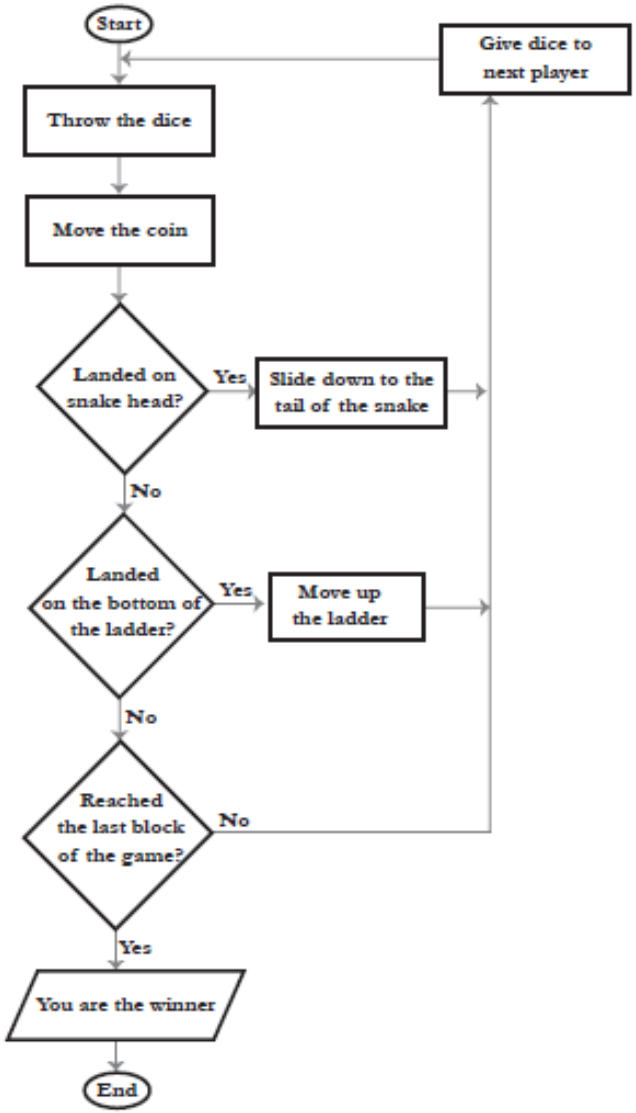
**Example:** Convert Temperature from Fahrenheit (°F) to Celsius (°C)



# Exercise

- Flowchart for computing factorial N (N!)
- Flowchart for finding area of a rectangle whose length and breadth are given
- Snakes and Ladder game
- How to find profit or loss.


This is a sequence.
Start
This is a decision structure.
Read Cost price (CP)
Read Selling price (SP)
Is SP > CP ?
Yes
Profit = SP - CP
No
Loss = CP - SP
Print Loss
Print Profit
End


Start
Give dice to next player
Throw the dice
Move the coin
Landed on snake head?
Yes
Slide down to the tail of the snake
No
Landed on the bottom of the ladder?
Yes
Move up the ladder
No
Reached the last block of the game?
No
Yes
You are the winner
End