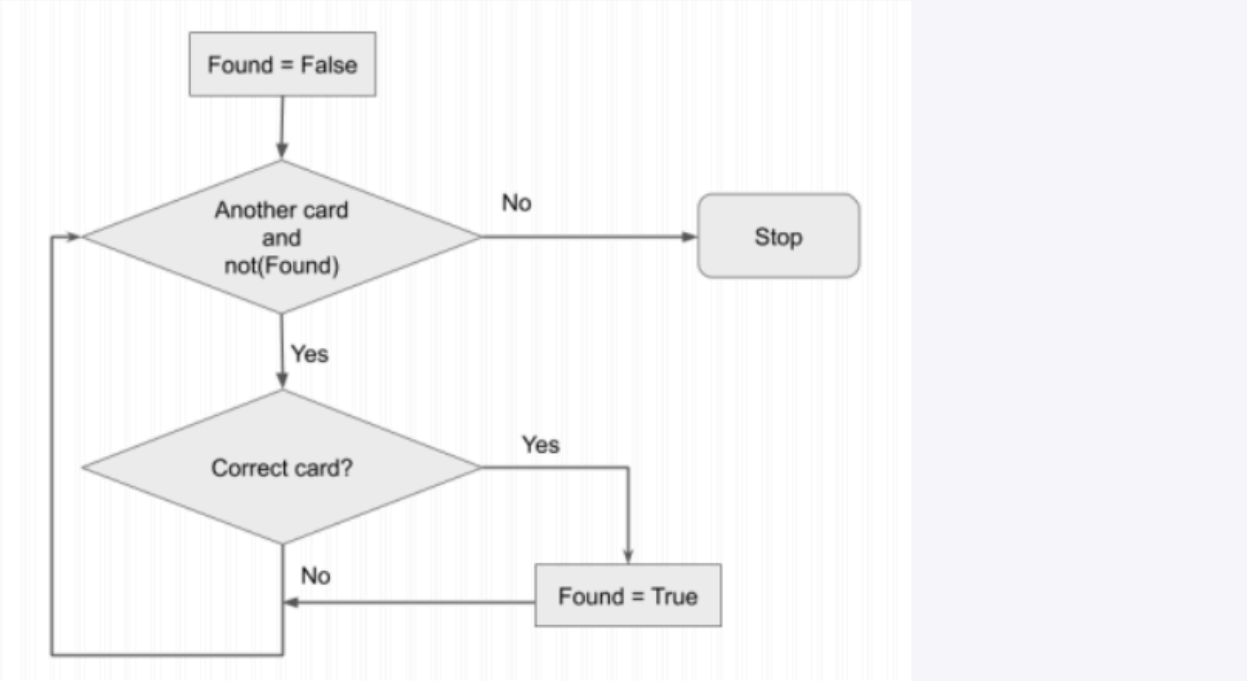
Week 2 : Iteration , Filtering , Selection , Pseudocode , Finding max and min , AND operator

Lecture 1 : Conditional termination in Iteration

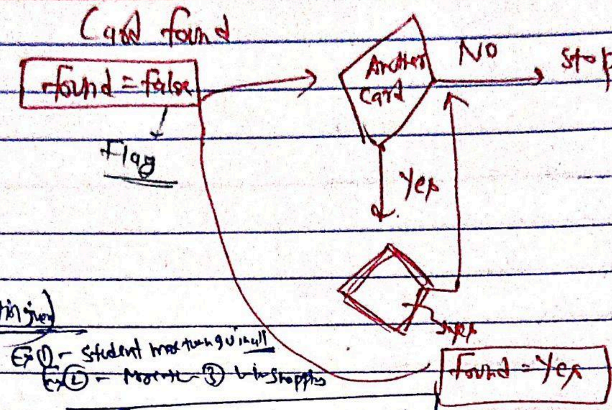
Conditional termination in iteration means **stopping a loop when a specific condition is met**, instead of running it for a fixed number of times.

Definition

Conditional termination occurs when **a loop ends based on a logical condition** — usually using statements like `if`, `break`, or `return`.



*



Conditional Termination in Iteration

↓

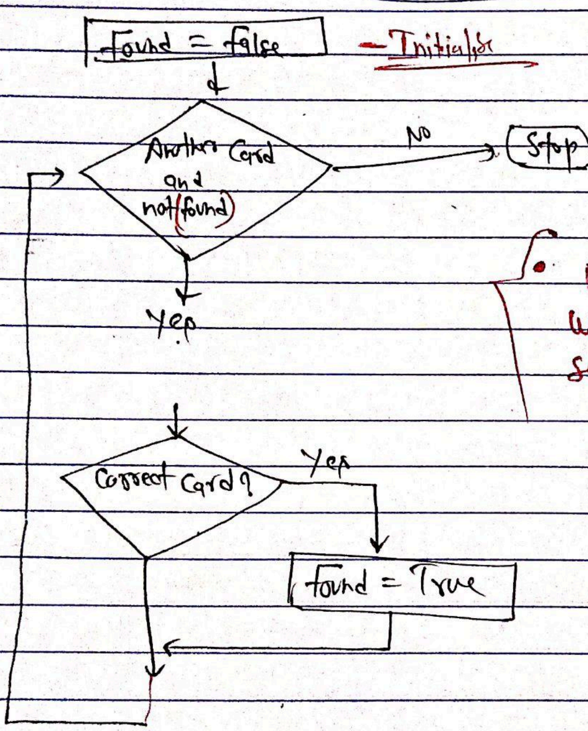
(remove card of condition given)

Ex① – Student most tough in all

Ex② – Month ③ Bus stopping

Ex3 # 3 sentence, 1 verb

*



• When we are searching and we have found what we were searching for. We can stop.

*. Conditional Termation in Iteration :- ①. Temporary Variable = T/F — Flag

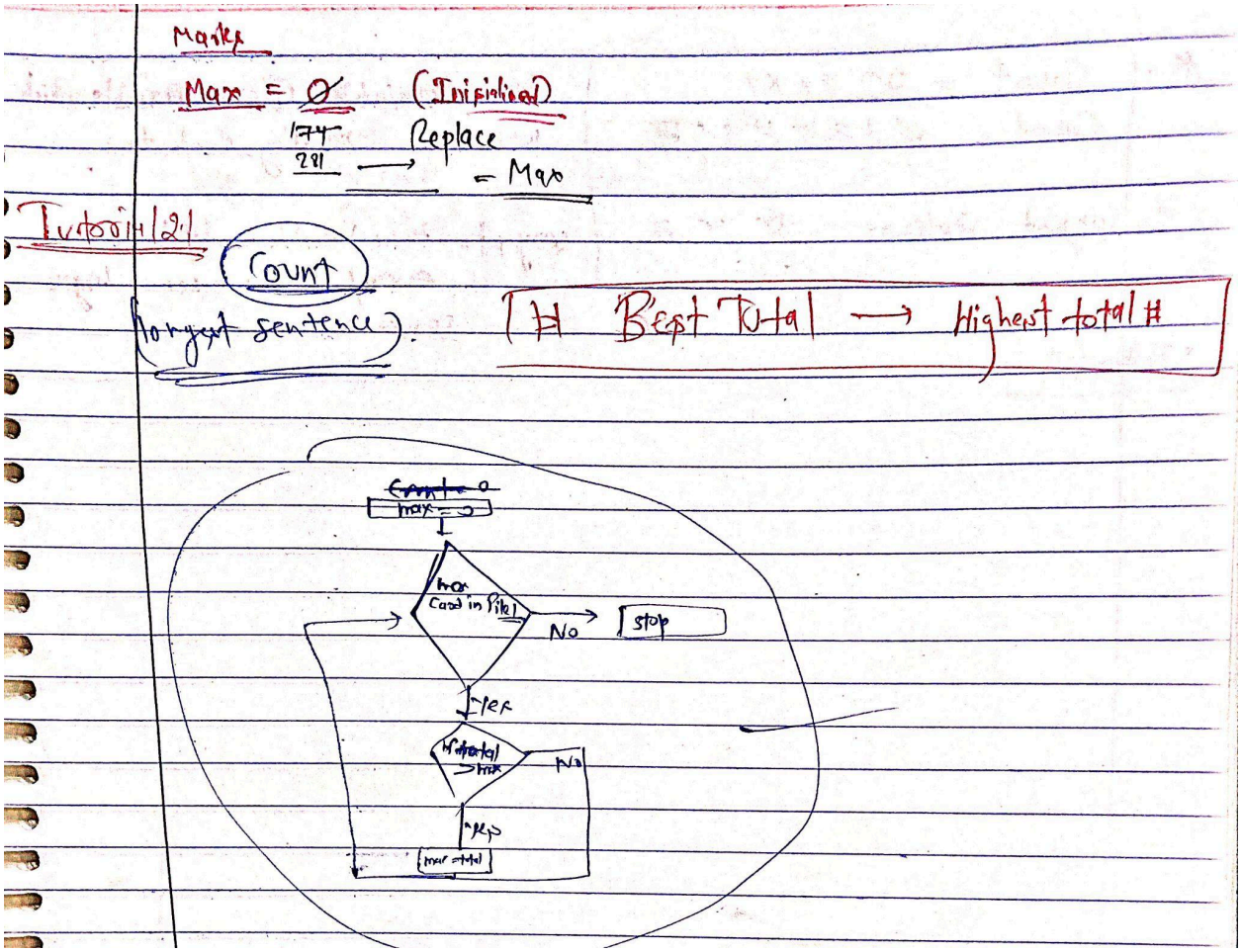
②. Check Condition → Decision → No

↓ Yes

▭ Processing

QN : 2,3

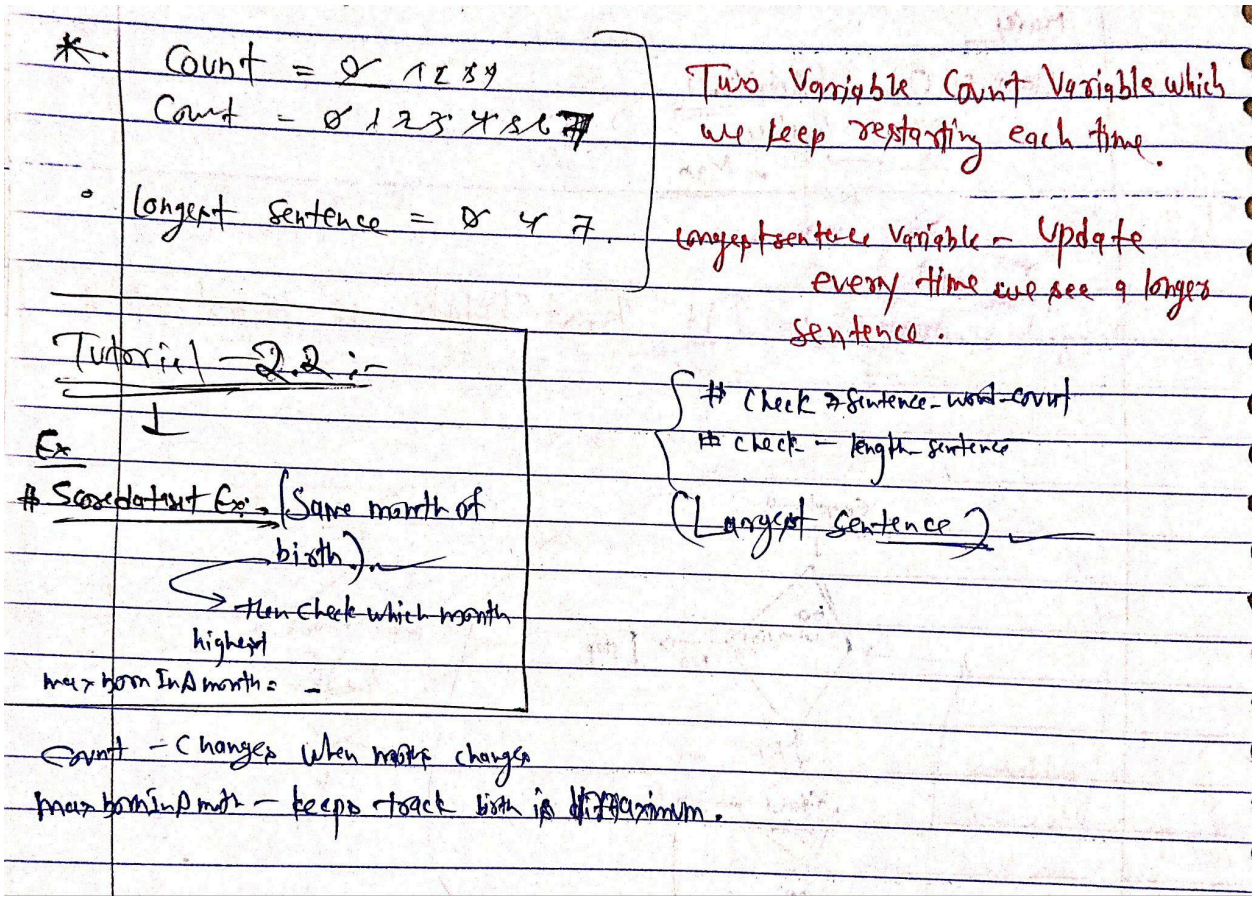
Lecture 2 : Local operations and max in single iteration (Part 1)



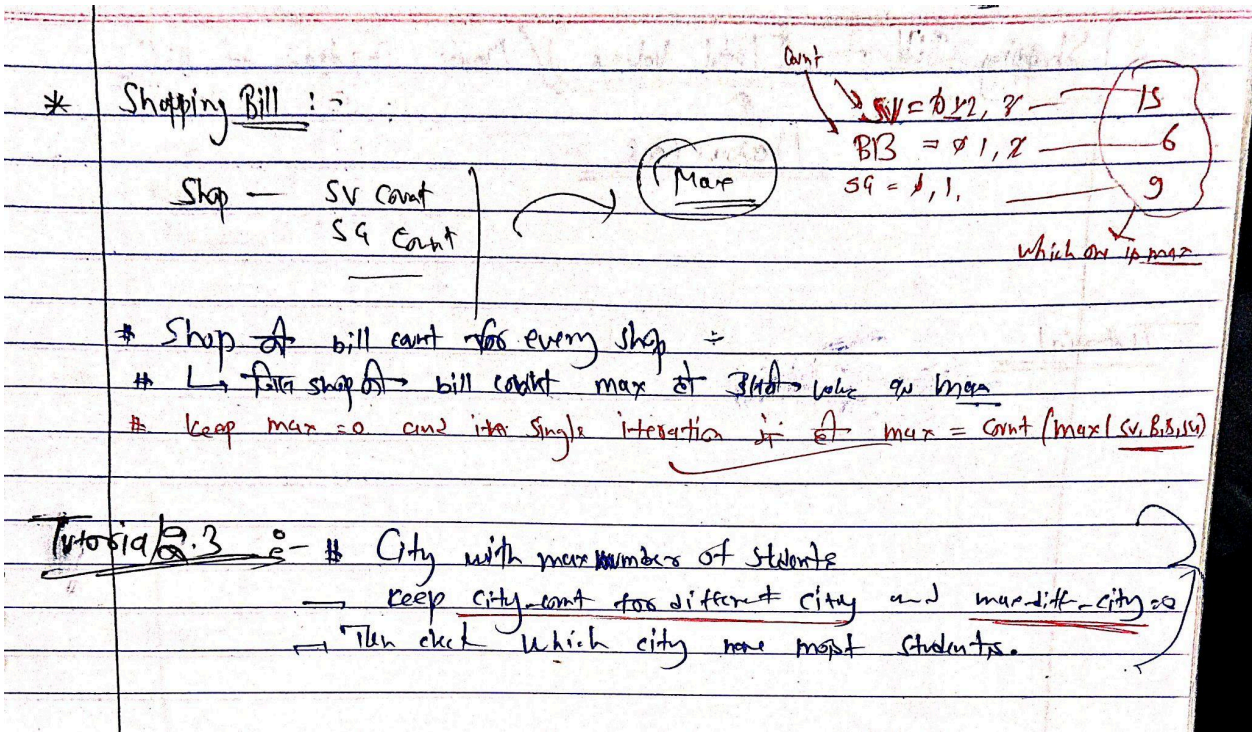
In Tutorial 2.1 Same for Largest total_amount in Shopping Bill dataset

QN : 1

Lecture 3 : Local operations and max in single iteration (Part 2)

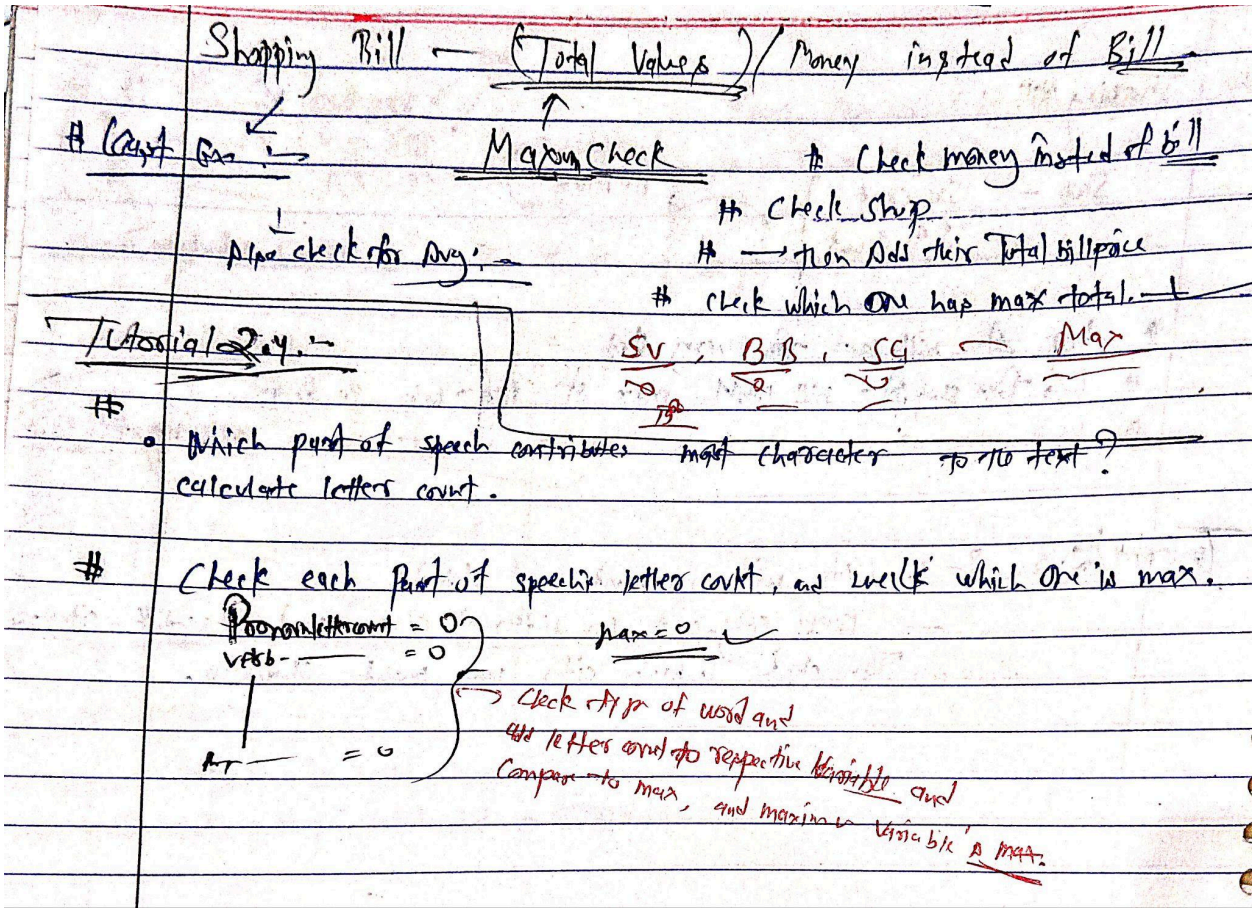


Lecture 4 : Local operations and max in single iteration (Part 3)



QN : 2,3

Lecture 5 : Local operations and max in single iteration (Part 4)



QN : 3

Lecture 6 : Max in a single iteration and max in two iterations (non-nested)

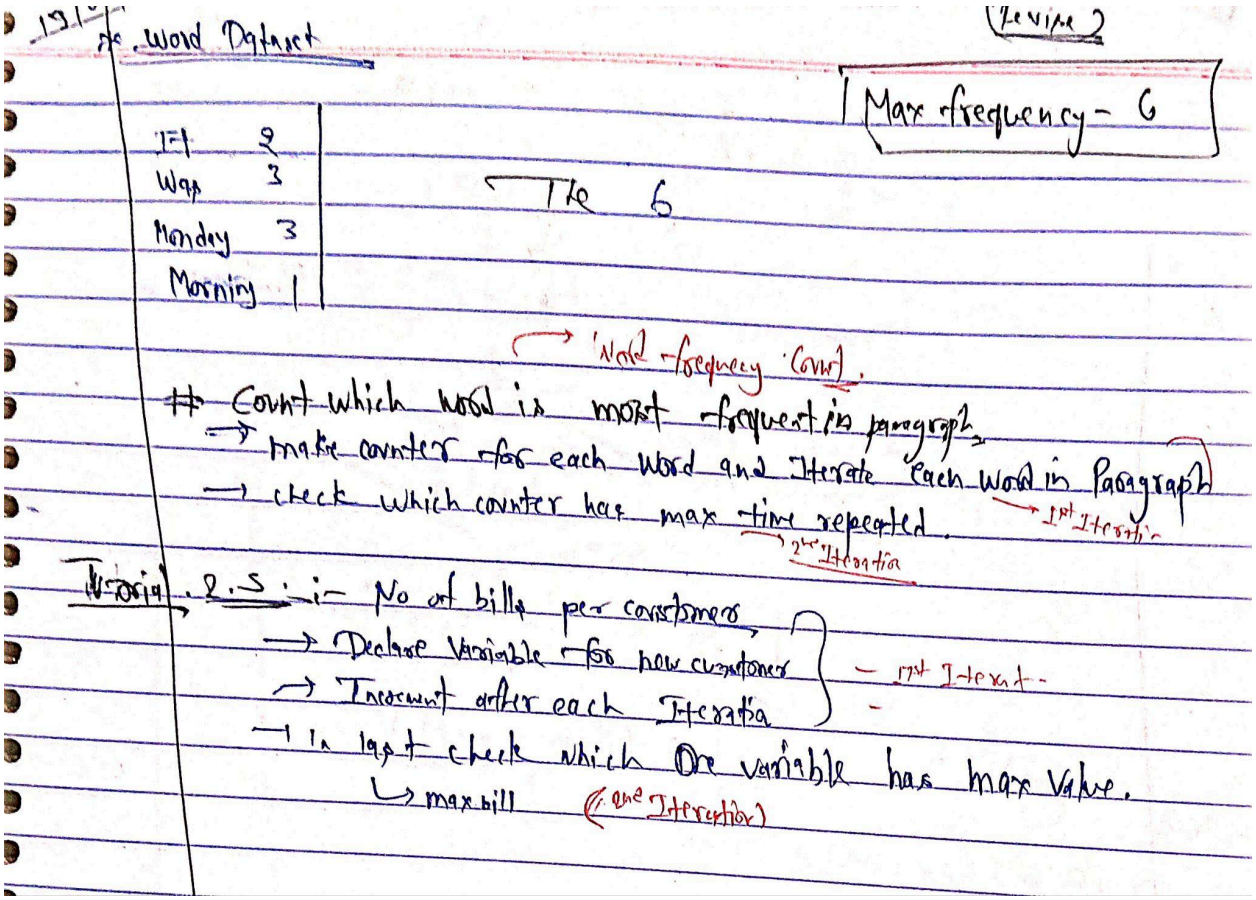
It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head Master with his thin long cane....

It	2	specially	1	get	1	building	1
was	3	unpleasant	1	into	1	fire-eyed	1
Monday	3	in	1	mood	1	Veda-nayagam	1
morning	1	the	6	work	1		
Swaminathan	1	calendar	1	discipline	1	class-teacher	1
reluctant	1	After	1	shuddered	1		
to	2	delicious	1	at	1	Head	1
open	1	freedom	1	very	1	Master	1
his	2	of	3	thought	1	with	1
eyes	1	Saturday	1	school	1	thin	1
He	2	and	3	that	1	long	1
considered	1	Sunday	1	dismal	1	cane	1
		difficult	1	yellow	1		

It was Monday morning. Swaminathan was reluctant to open his eyes. He considered Monday specially unpleasant in the calendar. After the delicious freedom of Saturday and Sunday, it was difficult to get into the Monday mood of work and discipline. He shuddered at the very thought of school: that dismal yellow building; the fire-eyed Vedanayagam, his class-teacher; and the Head Master with his thin long cane....

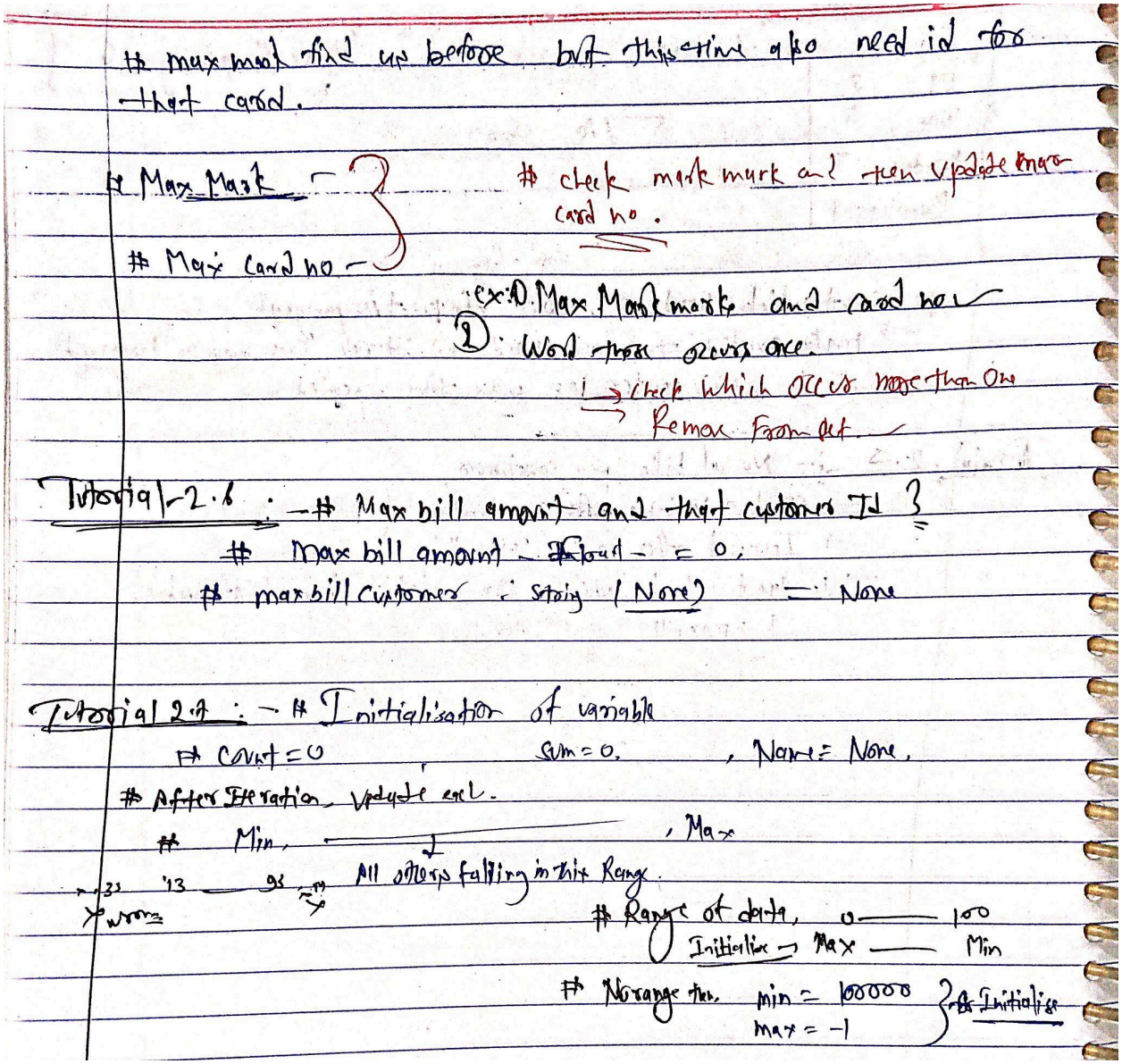
Max-frequency	6

It	2	specially	1	get	1	building	1
was	3	unpleasant	1	into	1	fire-eyed	1
Monday	3	in	1	mood	1	Veda-nayagam	1
morning	1	the	6	work	1		
Swaminathan	1	calendar	1	discipline	1	class-teacher	1
reluctant	1	After	1	shuddered	1		
to	2	delicious	1	at	1	Head	1
open	1	freedom	1	very	1	Master	1
his	2	of	3	thought	1	with	1
eyes	1	Saturday	1	school	1	thin	1
He	2	and	3	that	1	long	1
considered	1	Sunday	1	dismal	1	cane	1
		difficult	1	yellow	1		


Revise 2
Word Dataset
It 2
Was 3
Monday 3
Morning 1
The 6
Max frequency - 6
→ Word frequency count.
Count which word is most frequent in paragraph.
→ make counter for each word and Iterate each word in Paragraph
→ check which counter has max time repeated.
1st Iteration
2nd Iteration
Variable : i- No of bills per customers
→ Declare Variable for new customer
→ Increment after each Iteration
- 1st Iteration
→ In last check which one variable has max value.
↳ max bill (one Iteration)

QN 1,2

Lecture 7 : Max in a single iteration without losing information and applications of frequency count

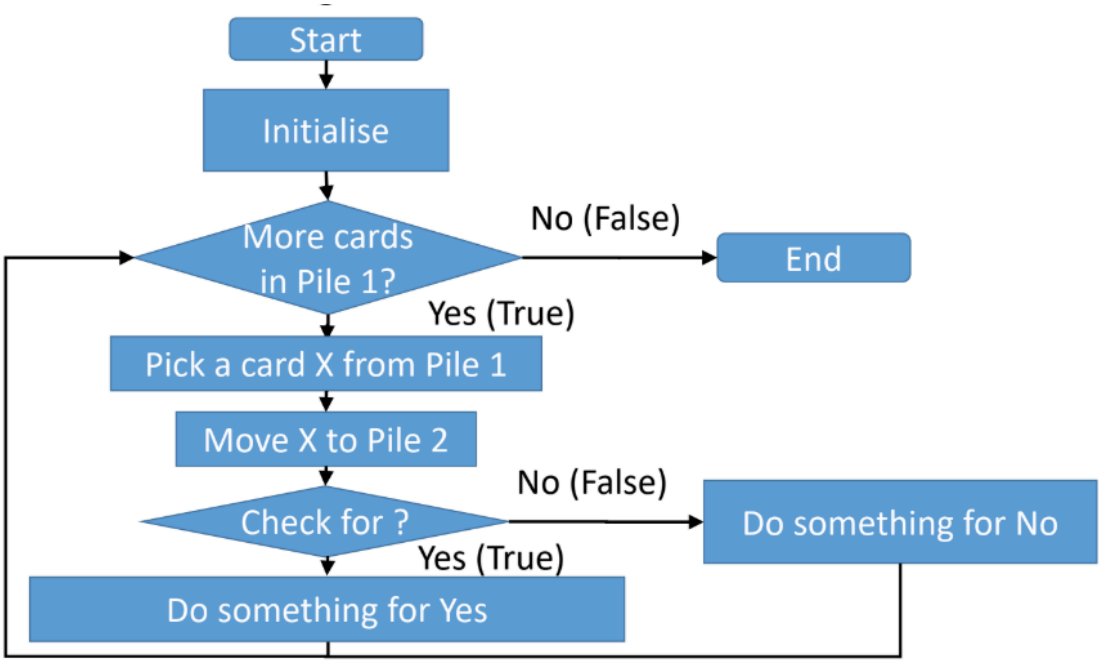


QN : 3

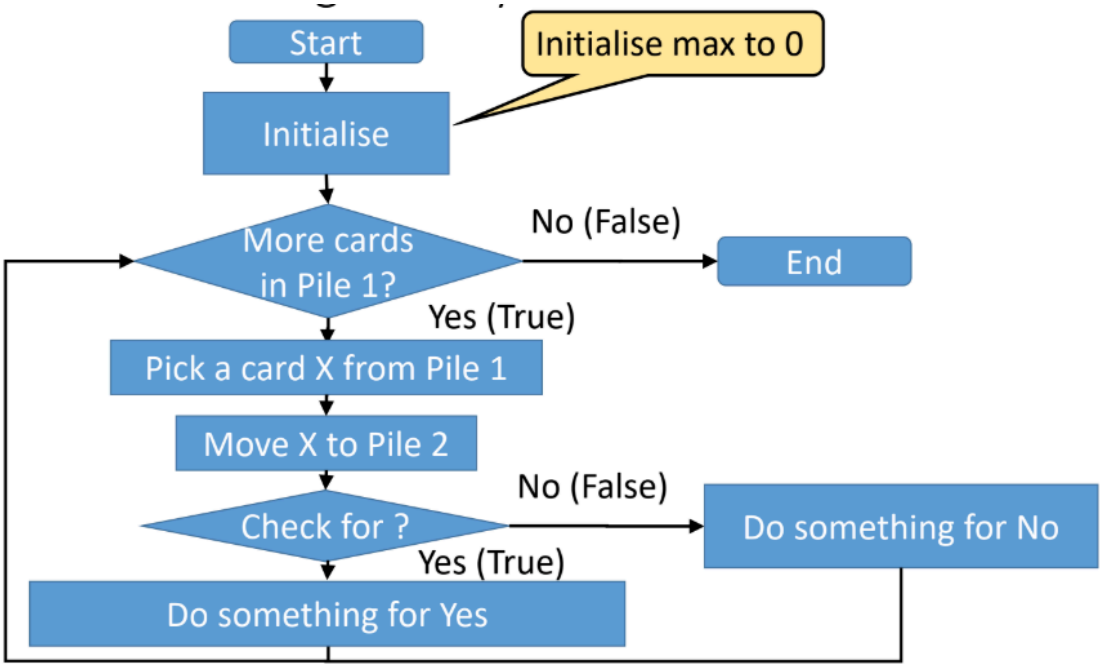
Lecture 8 : Flowchart for max marks

Flowchart for Max of Maths

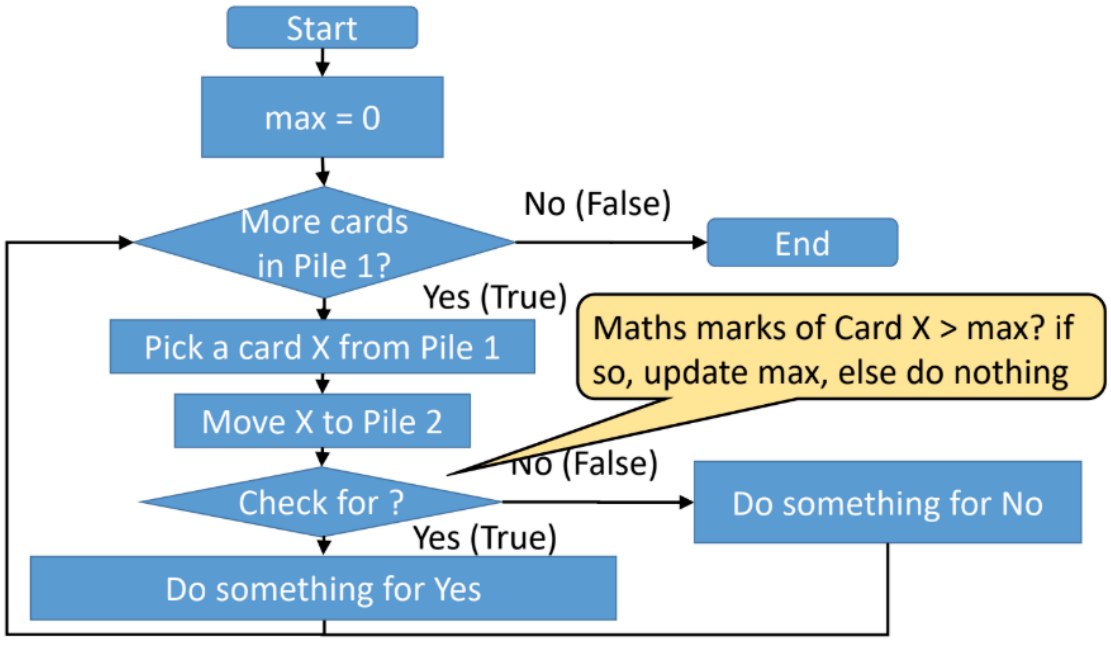
Iteration with filtering



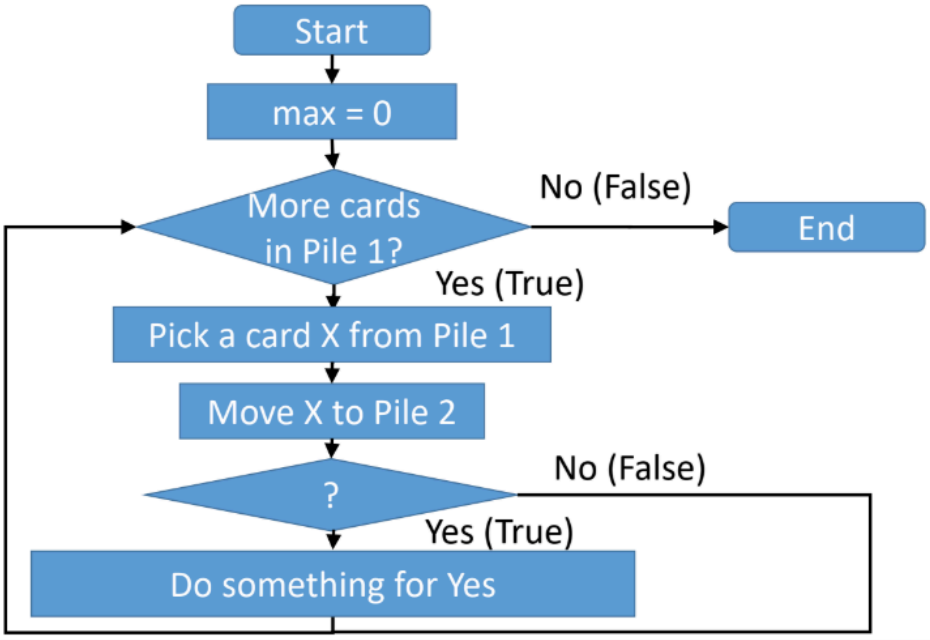
Iteration with filtering: modify for max



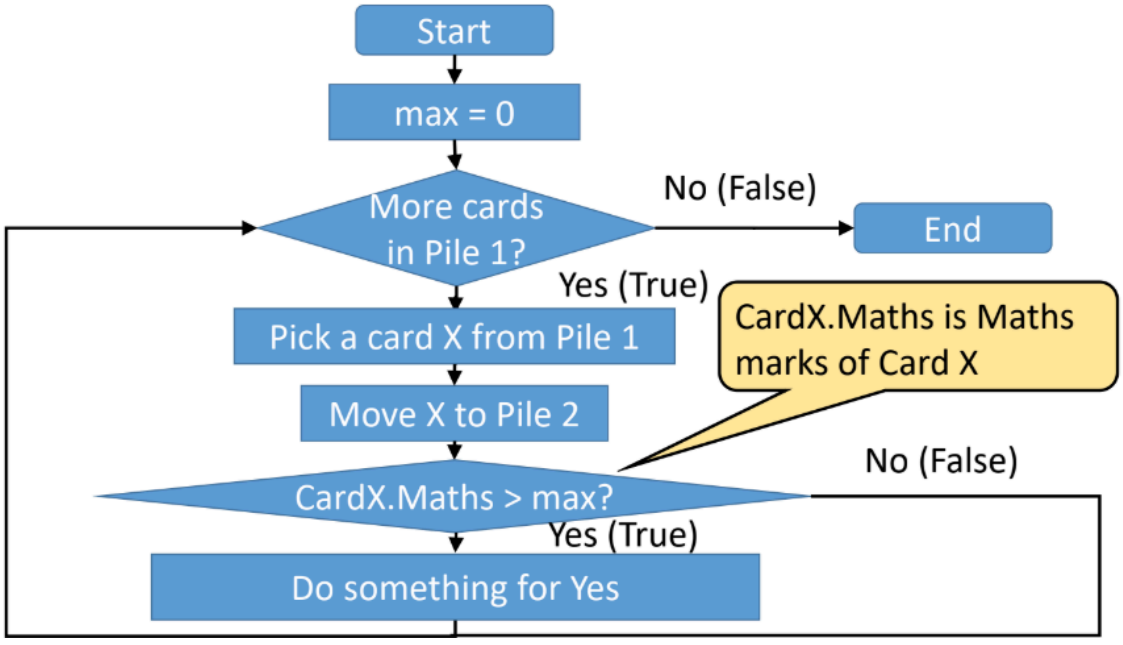
2.



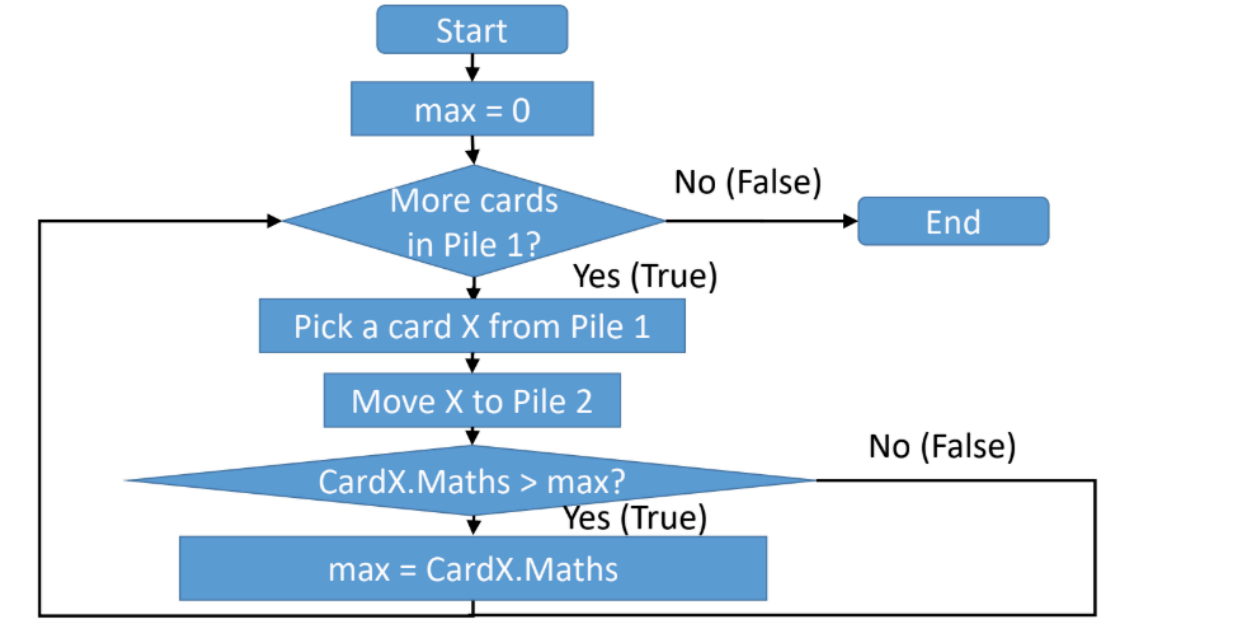
3.


Start
max = 0
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
?
No (False)
Yes (True)
Do something for Yes

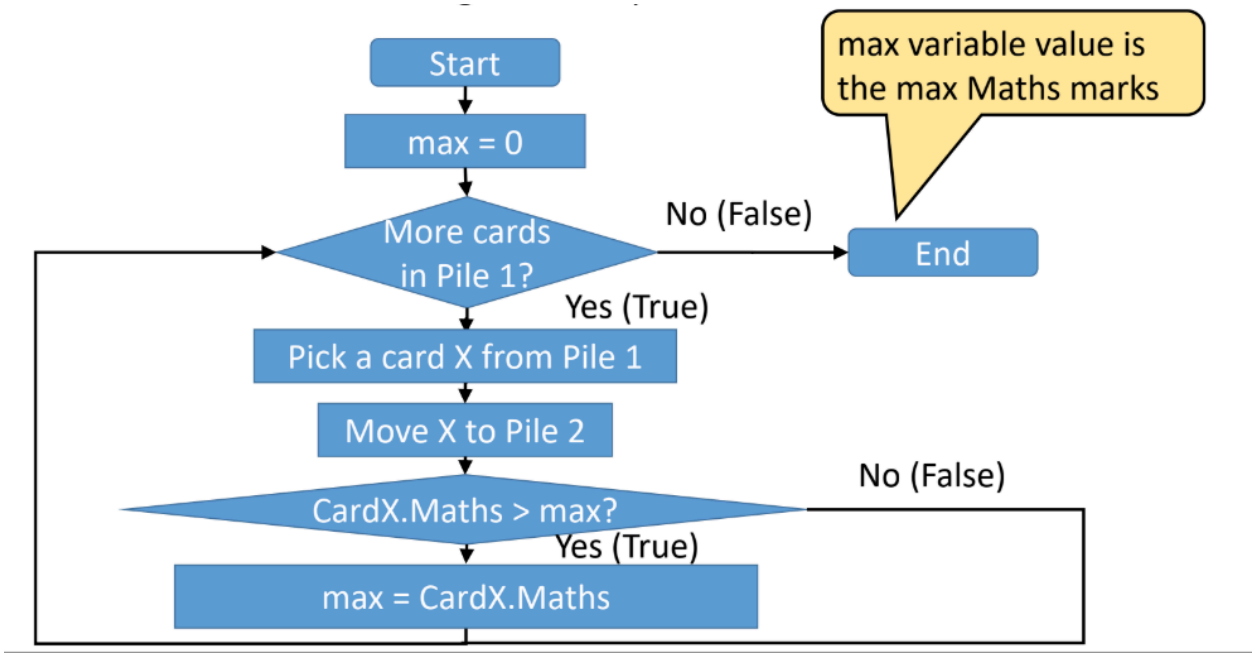
4.


Start
max = 0
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
CardX.Maths is Maths marks of Card X
Move X to Pile 2
CardX.Maths > max?
No (False)
Yes (True)
Do something for Yes

5.

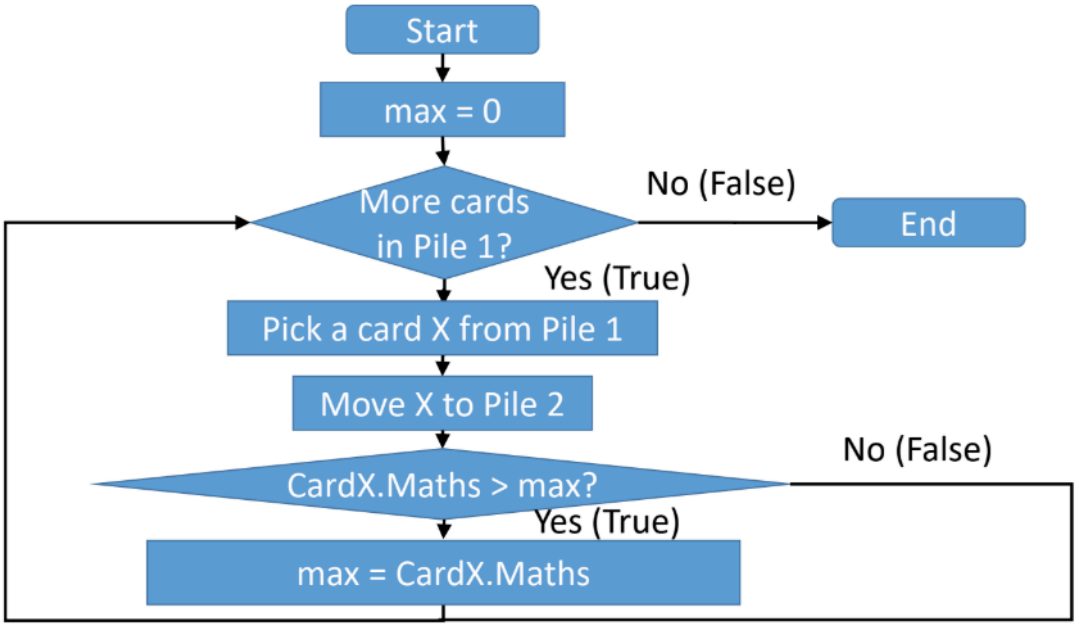

Start
max = 0
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
Yes (True)
max = CardX.Maths

6.

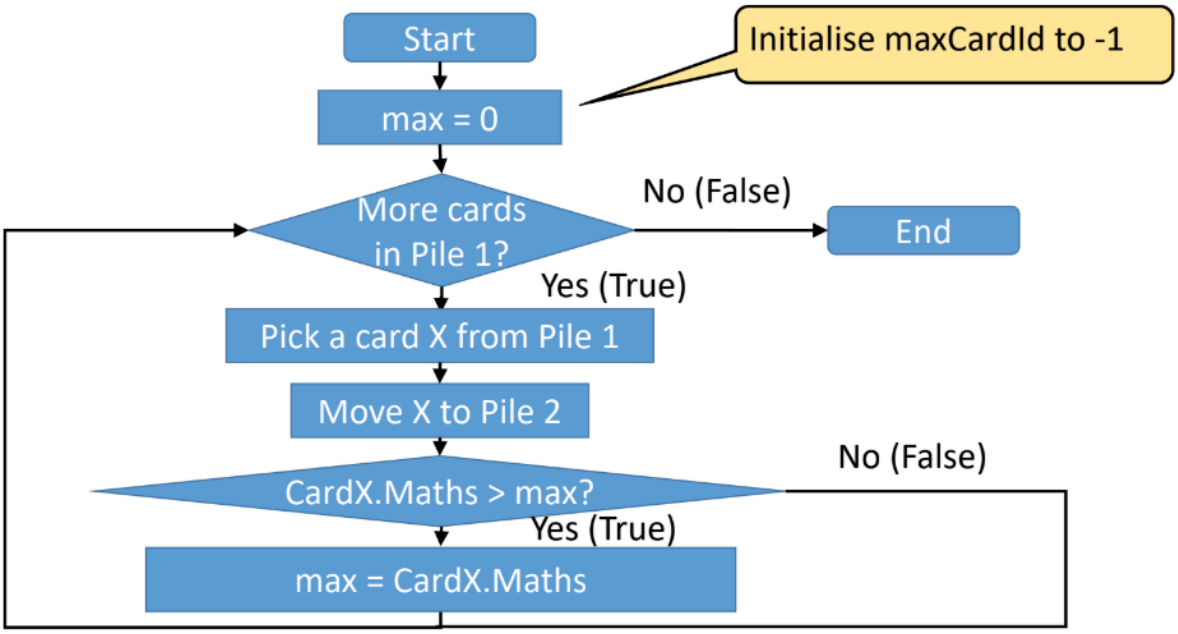

max variable value is the max Maths marks
Start
max = 0
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
Yes (True)
max = CardX.Maths

Flowchart for Max of Maths Marks; keep track of the card which has the max marks

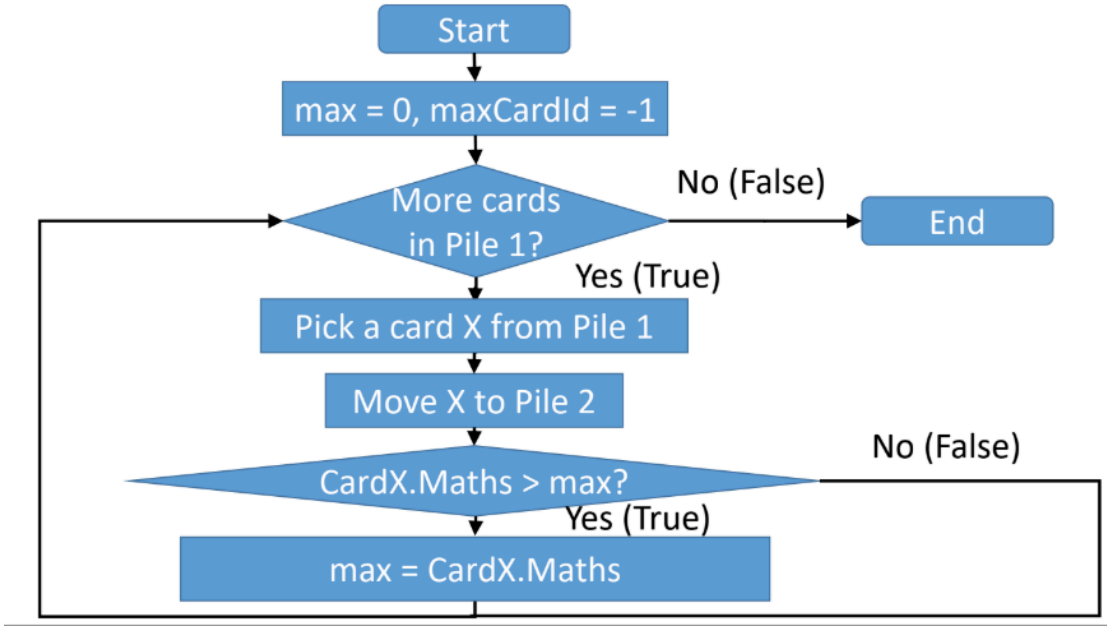
Max of Maths marks: keep track of card



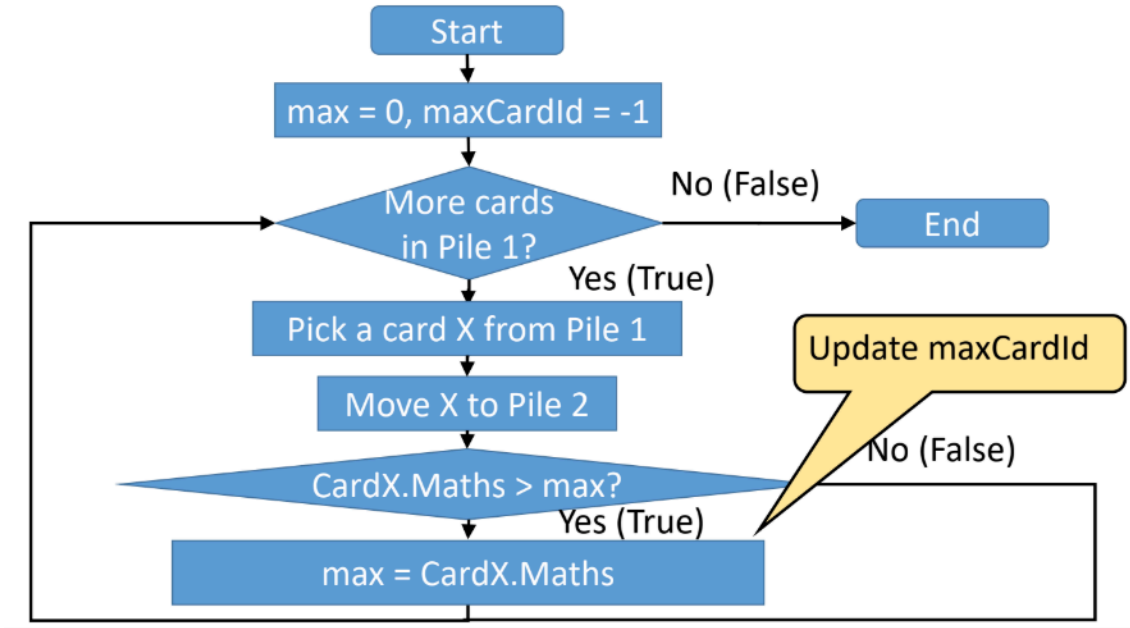
2.



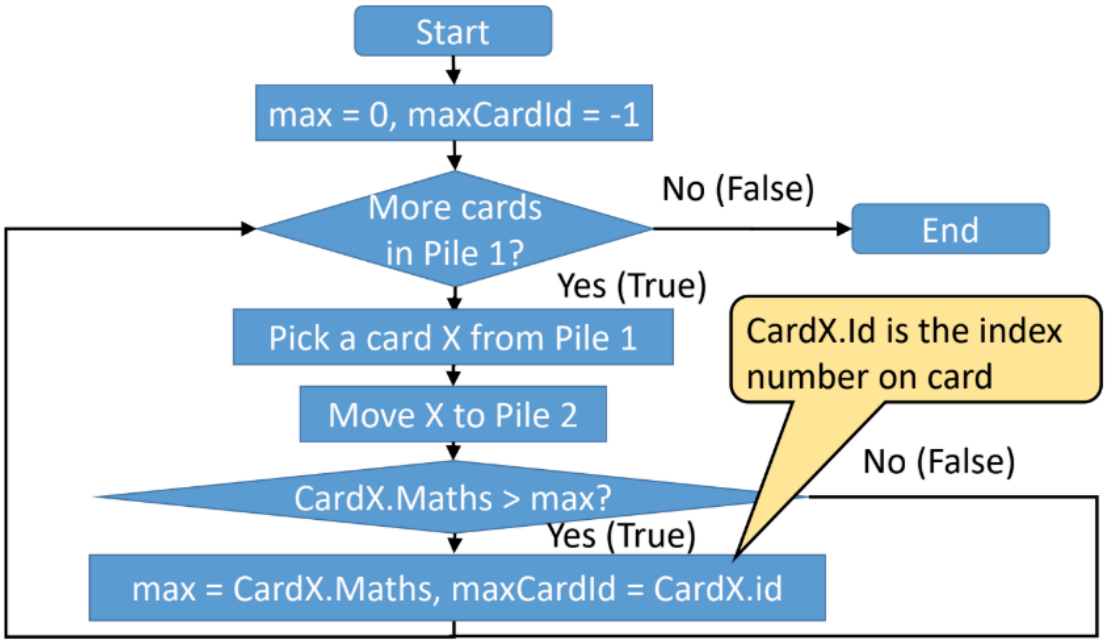
3.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
Yes (True)
max = CardX.Maths

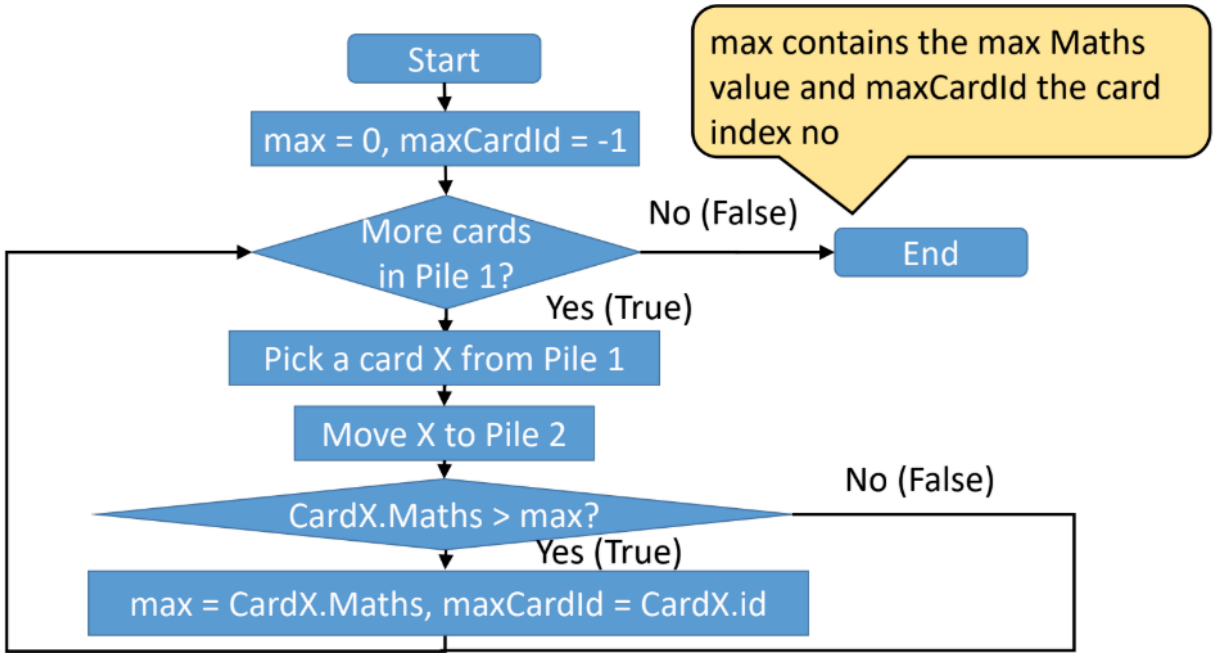
4.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Update maxCardId
Move X to Pile 2
CardX.Maths > max?
No (False)
Yes (True)
max = CardX.Maths

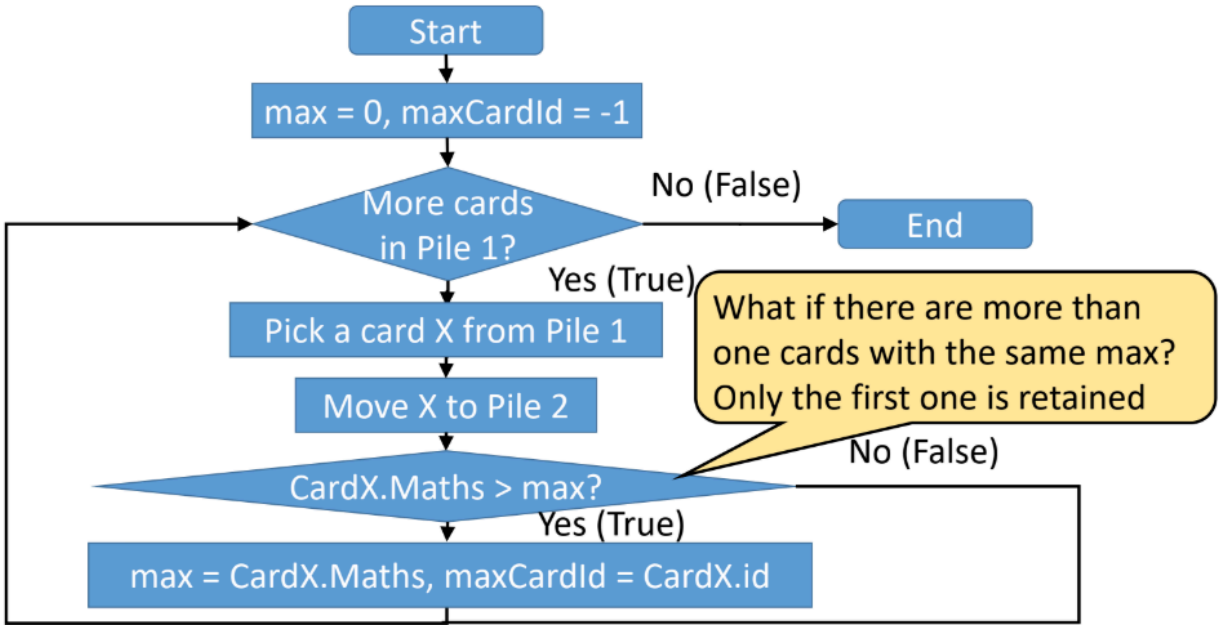
5.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
CardX.Id is the index number on card
Move X to Pile 2
No (False)
CardX.Maths > max?
Yes (True)
max = CardX.Maths, maxCardId = CardX.id

6.

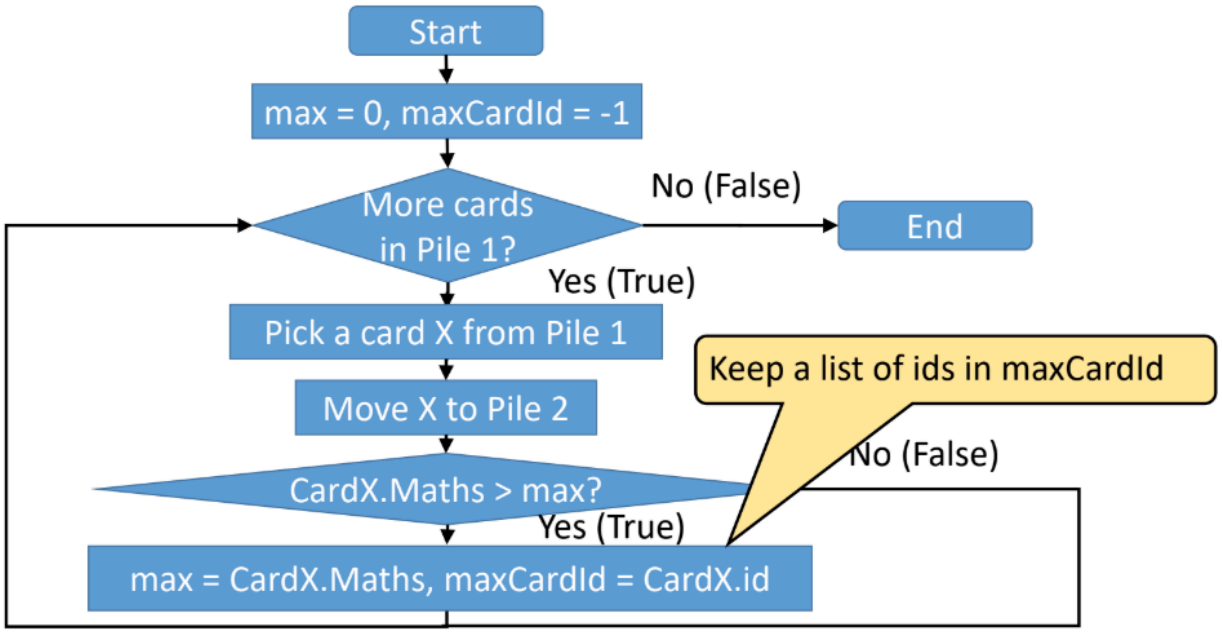

max contains the max Maths value and maxCardId the card index no
Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
No (False)
CardX.Maths > max?
Yes (True)
max = CardX.Maths, maxCardId = CardX.id

Max of Maths marks: keep track of card - PROBLEM !

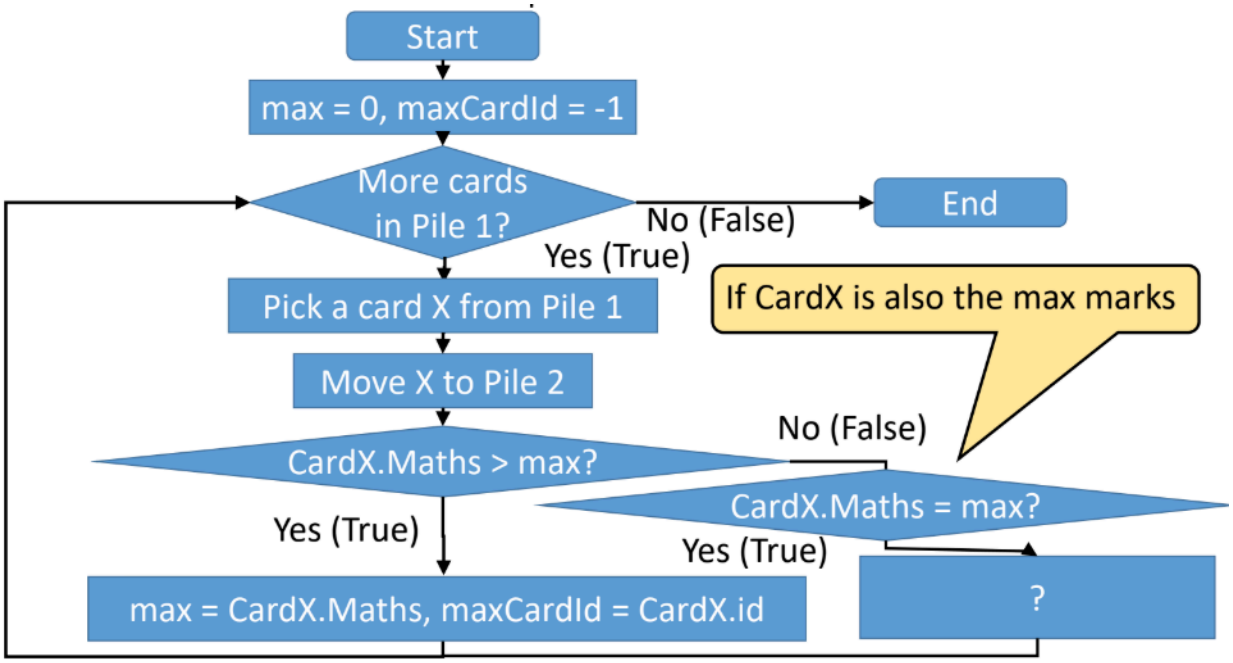


Max of Maths marks: keep track of card - SOLUTION !

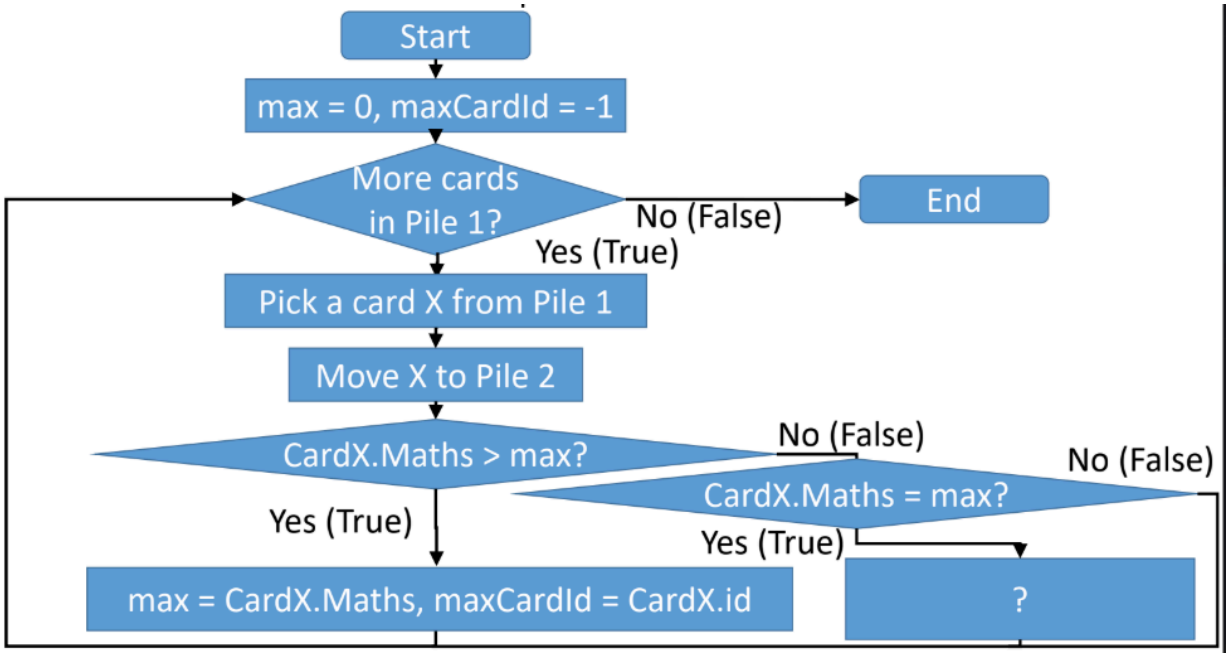
1.



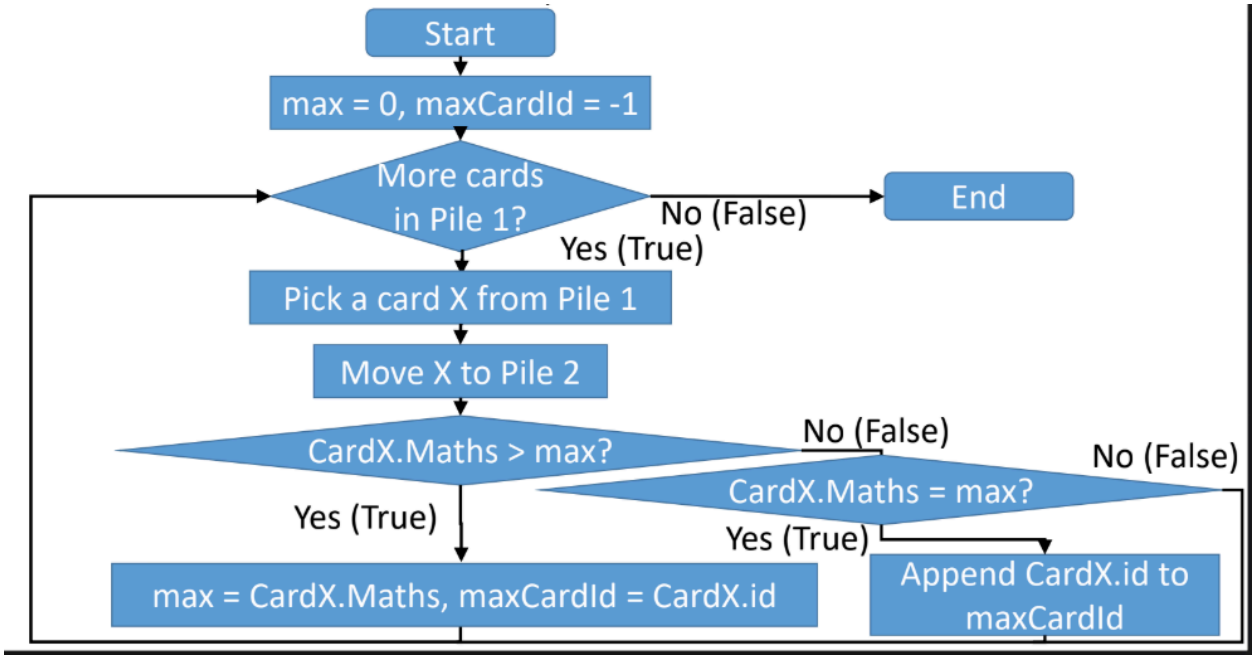
2.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
If CardX is also the max marks
Move X to Pile 2
No (False)
CardX.Maths > max?
CardX.Maths = max?
Yes (True)
Yes (True)
?
max = CardX.Maths, maxCardId = CardX.id

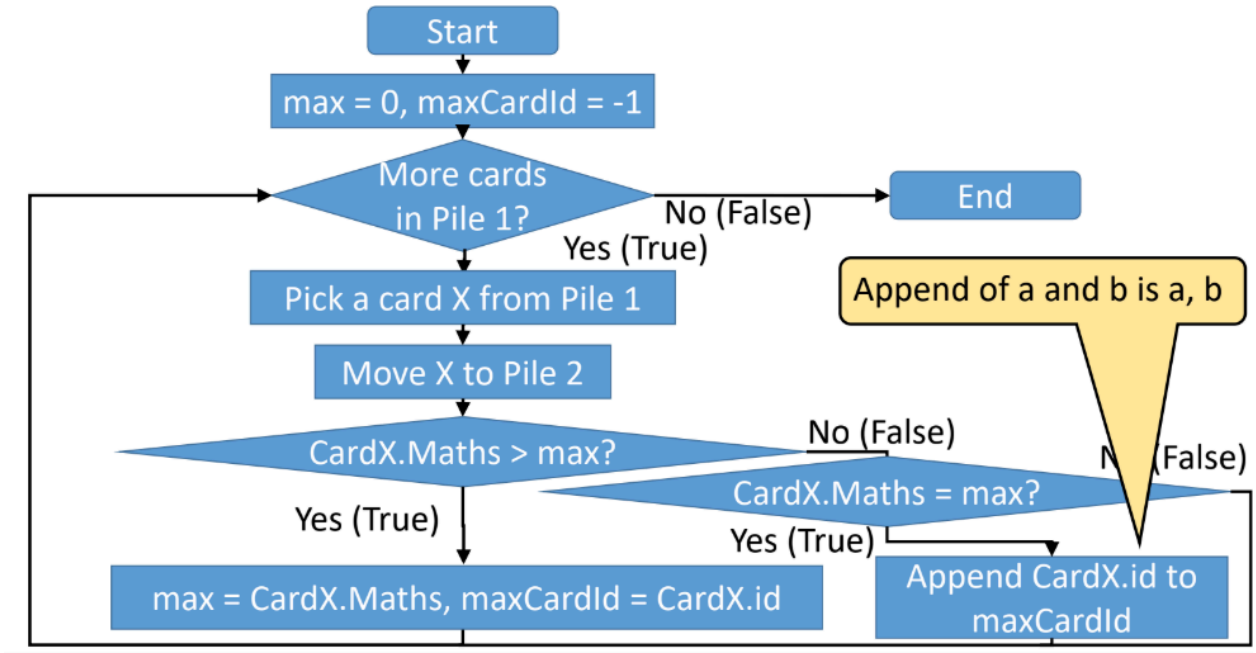
3.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
No (False)
CardX.Maths > max?
No (False)
CardX.Maths = max?
Yes (True)
Yes (True)
?
max = CardX.Maths, maxCardId = CardX.id

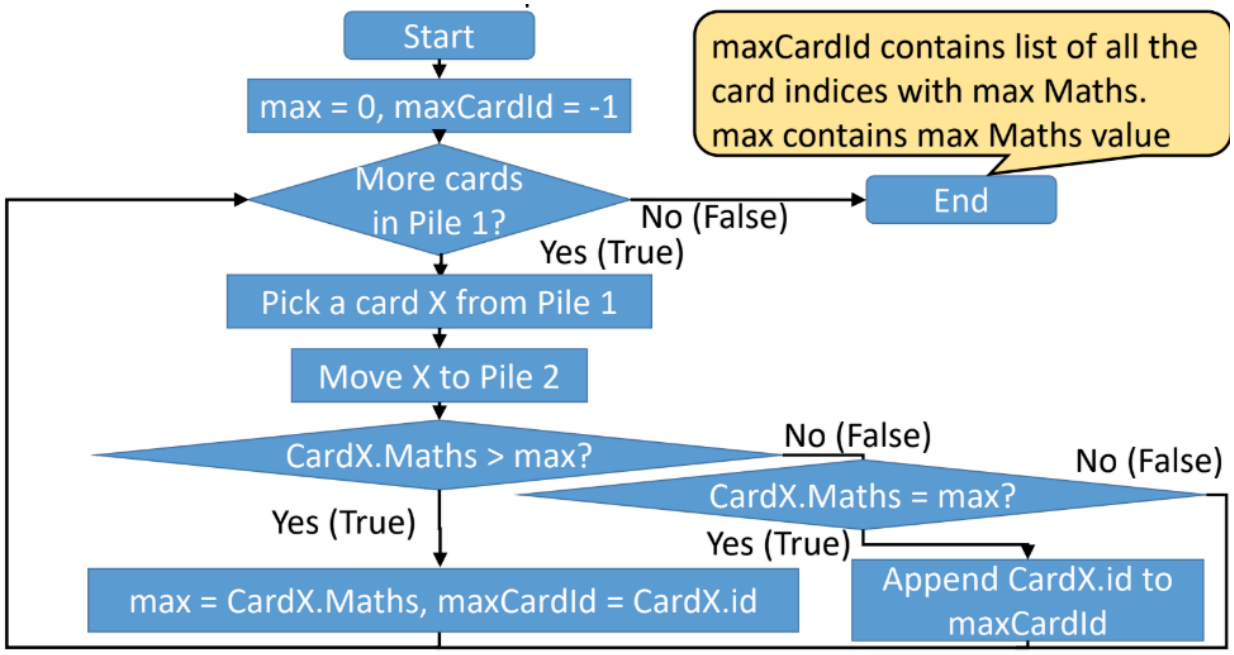
4.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
CardX.Maths = max?
No (False)
Yes (True)
Yes (True)
max = CardX.Maths, maxCardId = CardX.id
Append CardX.id to maxCardId

5.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
End
Yes (True)
Append of a and b is a, b
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
CardX.Maths = max?
Yes (True)
Yes (True)
max = CardX.Maths, maxCardId = CardX.id
Append CardX.id to maxCardId

6.


Start
max = 0, maxCardId = -1
More cards in Pile 1?
No (False)
Yes (True)
maxCardId contains list of all the card indices with max Maths. max contains max Maths value
End
Pick a card X from Pile 1
Move X to Pile 2
CardX.Maths > max?
No (False)
CardX.Maths = max?
No (False)
Yes (True)
Yes (True)
max = CardX.Maths, maxCardId = CardX.id
Append CardX.id to maxCardId

QN : 2,3,4

Lecture 9 : Introduction to pseudocodes

Pseudocode: From pictures to text

Flowchart

Definition

A **flowchart** is a **diagrammatic representation** of an algorithm or process.
It uses **symbols and arrows** to show the **flow of control** (the order of execution of steps).

A flowchart is a **picture** of how a program or algorithm works.

Purpose

- To **visualize** the logic of an algorithm before coding.
- To **understand** the flow of decisions and loops.
- To **communicate** ideas clearly with others.

Pseudocode

Definition

Pseudocode means *"false code"* — it looks like programming code but isn't written in any specific programming language.
It describes the **steps of an algorithm in plain English**, using logical structures (like loops, if-else, etc.).

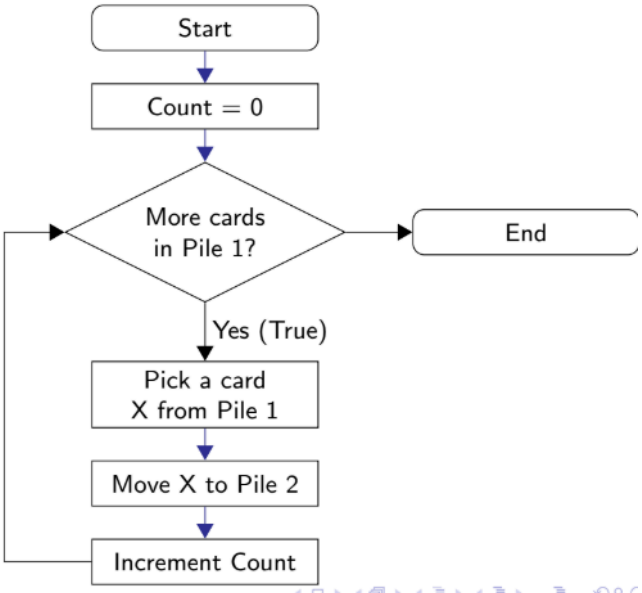
It helps programmers plan before writing real code.

Features

- Uses plain English + programming structure
- Easy to read and understand
- Focuses on *logic*, not syntax
- Can be converted easily into any programming language

Flowcharts

- Pictorial representation of computational process
  - Counting the number of cards
- Node types
  - Process
  - Decision
  - Terminal
- Arrows indicate operation flow



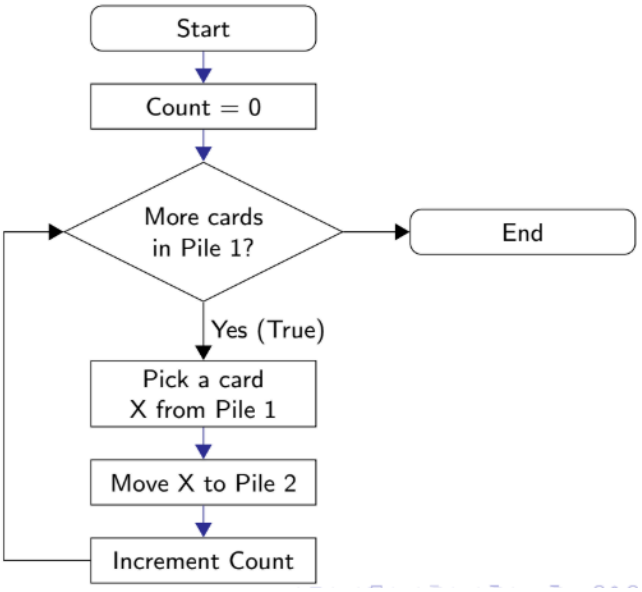
Pros and cons of flowcharts

Advantages

- Visual representation of computation
- Easy to understand

Disadvantages

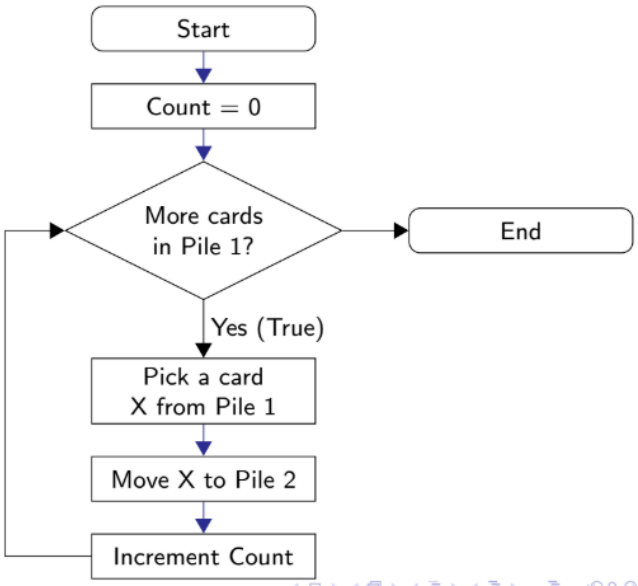
- Size: Complex processes generate large flowcharts
- Collaboration: Sharing pictures in editable format
- Versions: Compare changes between flowcharts



From pictures to text

Describe the process in words

Step 0 Start
Step 1 Initialize **Count** to 0
Step 2 Check cards in Pile 1
Step 3 If no more cards, go Step 8
Step 4 Pick a card **X** from Pile 1
Step 5 Move **X** to Pile 2
Step 6 Increment **Count**
Step 7 Go back to Step 2
Step 8 End



Programming language

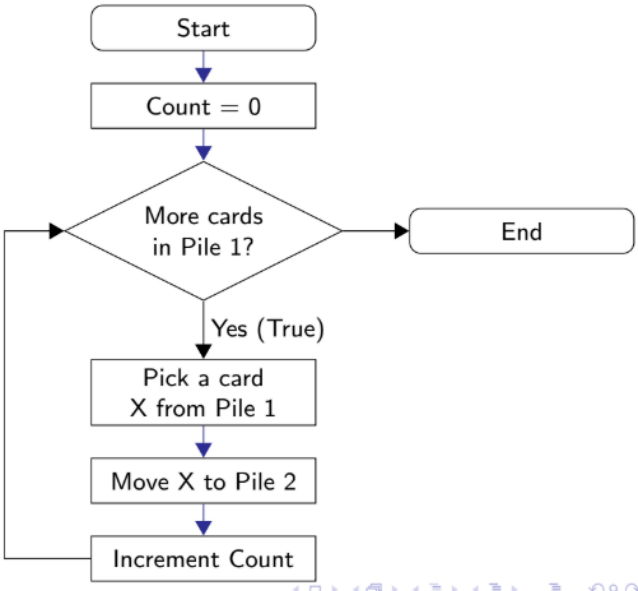
- Succinct notation for computational processes
- Better textual representation for

Conditional execution

Step 3 If no more cards, go to Step 8
Step 4 Pick a card **X** from Pile 1

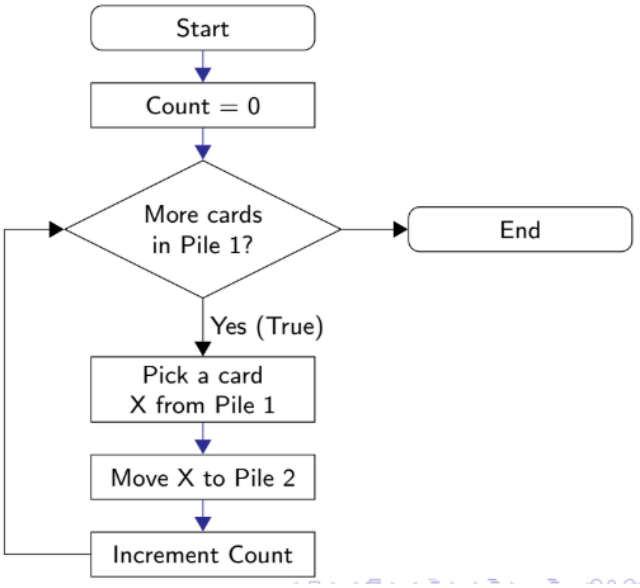
Repeated execution

Step 2 Check cards in Pile 1
⋮
Step 7 Go back to Step 2



Pseudocode

```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```

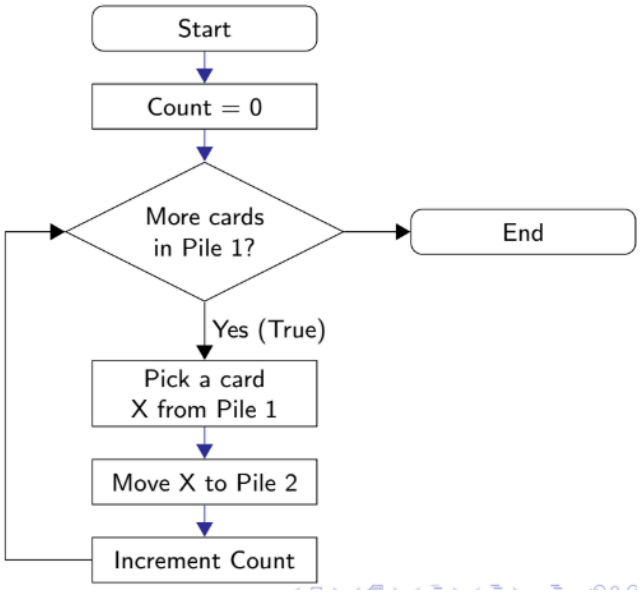


Explanation :

1.

```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```

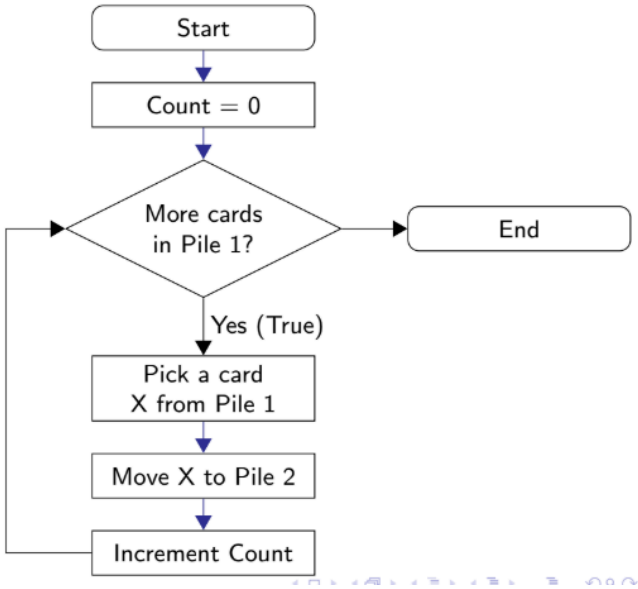
1. Assign a value to a variable



2.

```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```

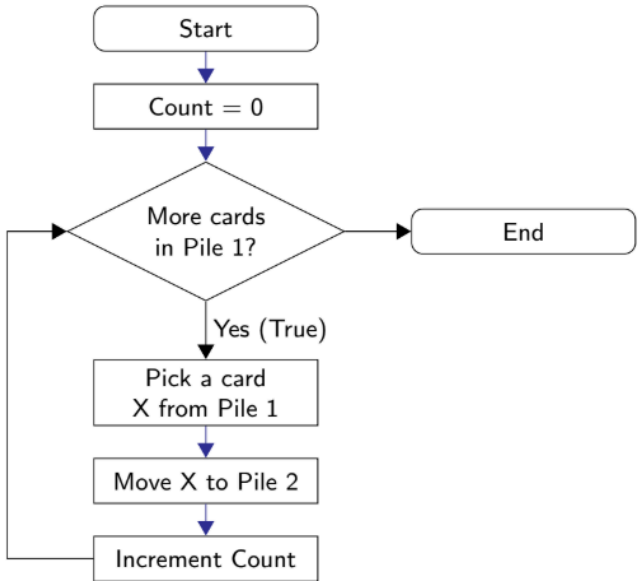
1. Assign a value to a variable
2. Repeat steps while condition holds



3.

```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```

1. Assign a value to a variable
2. Repeat steps while condition holds
3. Mark start and end of repeated block



Summary

- Flowcharts are easy to read, visual descriptions of procedures
- . . . but they are cumbersome, hard to share and edit
- Writing down steps in text is an alternative
- Tune the notation to capture standard features
  - Assigning values to variables
  - Conditional execution
  - Repeated execution

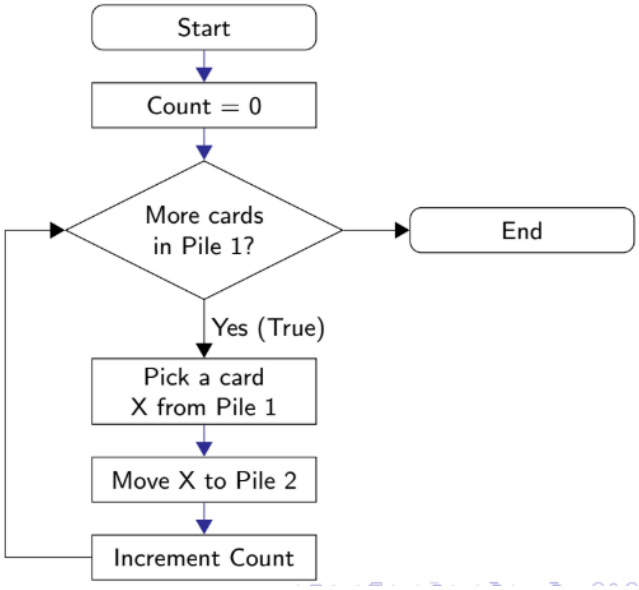
QN : 3

Lecture 10 : Pseudocode for iteration with filtering

Pseudocode: Iteration and Filtering

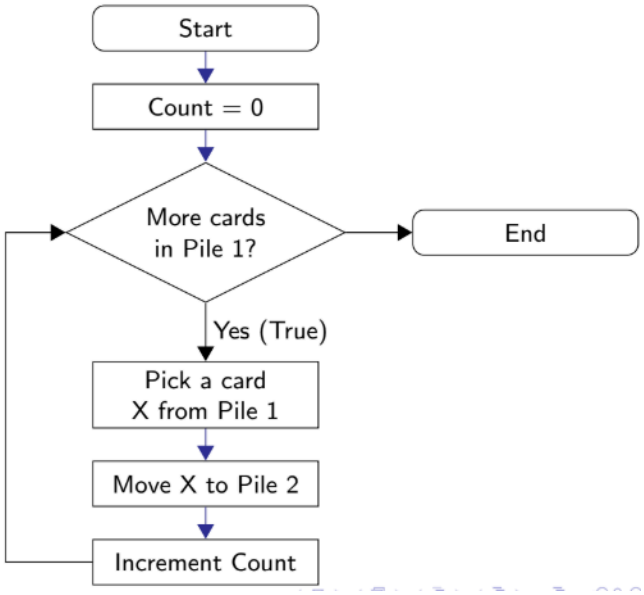
Counting cards

```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```




```
Start
Count = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  Increment Count
}
End
```

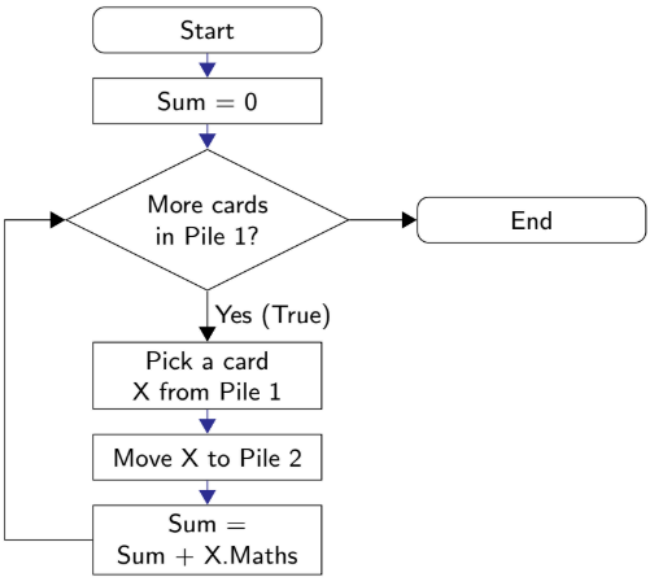
- Will dispense with Start and End, henceforth



Sum of Maths marks

Sum = 0
while (Pile 1 has more cards) {
  Pick a card **X** from Pile 1
  Move **X** to Pile 2
  Sum = **Sum** + **X**.Maths
}

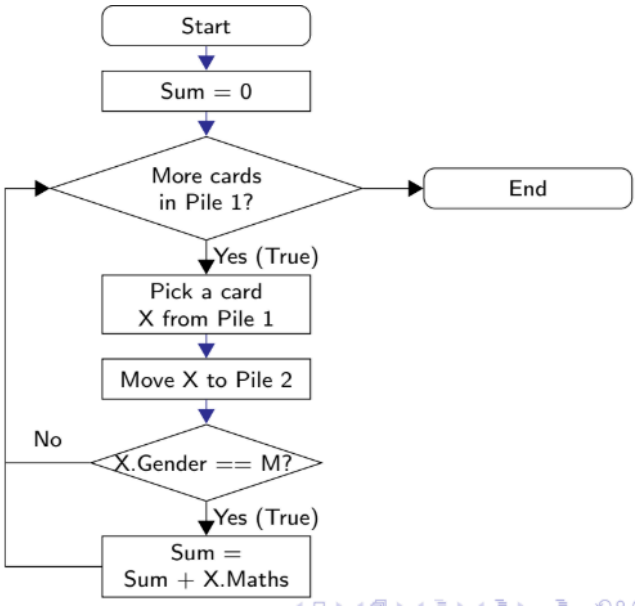
- Update **Sum** : assignment statement
  - **Sum** on right is current value
  - **Sum** on left is updated value
  - = is not mathematical equality!
- Increment: **Count** = **Count** + 1
- **X**.Maths : Maths marks in card **X**



Sum of Boys' Maths marks

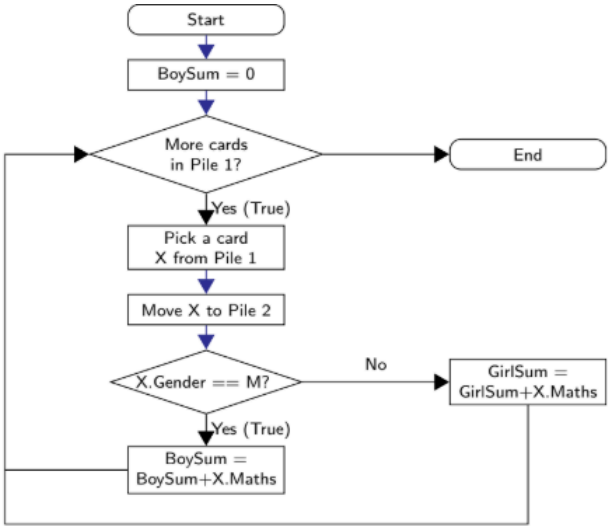
Sum = 0
while (Pile 1 has more cards) {
  Pick a card **X** from Pile 1
  Move **X** to Pile 2
  if (**X**.Gender == M) {
    Sum = **Sum** + **X**.Maths
  }
}

- Conditional execution, once
- Equality (==) vs assignment (=)



Sum of Boys' and Girls' Maths marks

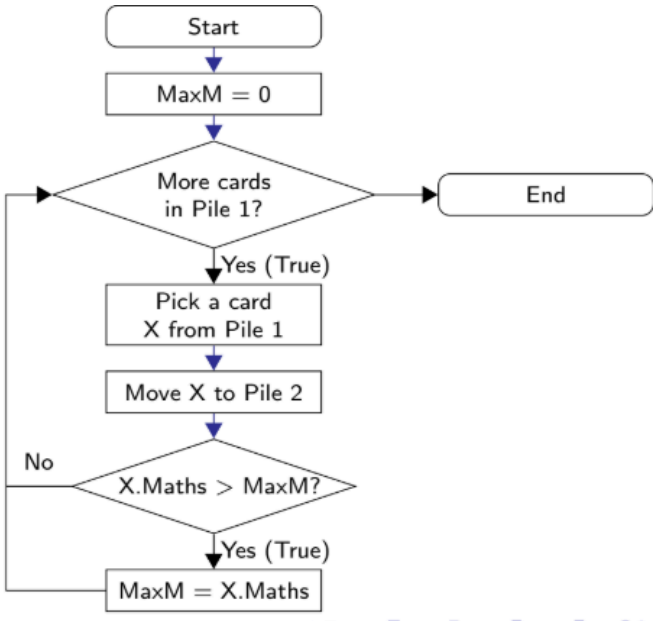
```
BoySum = 0
GirlSum = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Gender == M) {
    BoySum = BoySum + X.Maths
  }
  else {
    GirlSum = GirlSum + X.Maths
  }
}
```



Alternative branch for conditional

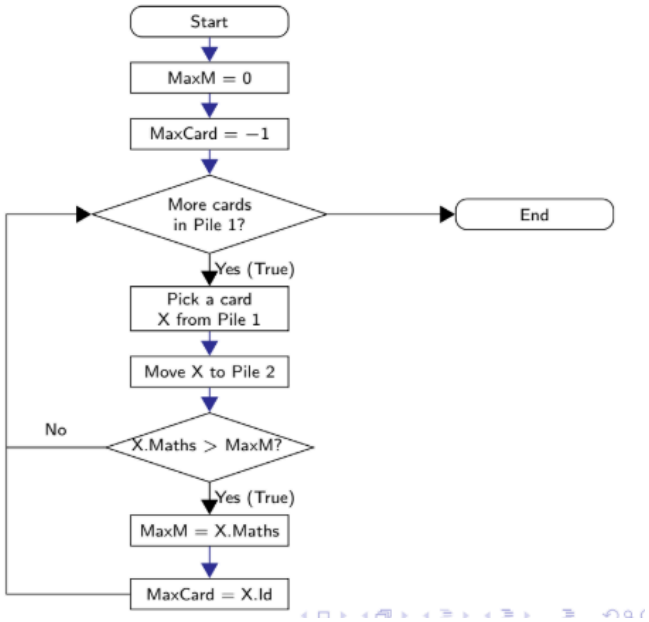
Finding the maximum Maths marks

```
MaxM = 0
while (Pile 1 has more cards) {
  Pick a card X from Pile 1
  Move X to Pile 2
  if (X.Maths > MaxM) {
    MaxM = X.Maths
  }
}
```



Finding the card with maximum Maths marks

MaxM = 0
MaxCard = −1
while (Pile 1 has more cards) {
  Pick a card **X** from Pile 1
  Move **X** to Pile 2
  if (**X**.Maths > **MaxM**) {
    MaxM = **X**.Maths
    MaxCard = **X**.Id
  }
}



Summary

- Assignment statement
  - **Count** = 0
  - **Sum** = **Sum** + **X**.Maths
- Conditional execution
  - Once
    - if (condition) { . . . }
    - if (condition) { . . . } else { . . . }
  - Repeatedly
    - while (condition) { . . . }
- Equality (==) vs assignment (=)
  - if (**X**.Gender == M)

QN : 2

Tutorial 2.8

```
count = 0
while (Pile 1 has more cards) {
        Read the top card X in Pile 1
        if (X.TownCity == "Chennai") {
                if (X.Gender == "Female") {
                        count = count + 1
                }
        }
        Move X to Pile 2
}
```

```
count = 0
while (Pile 1 has more cards) {
        Read the top card X in Pile 1
        if (X.TownCity == "Chennai" AND X.Gender == "Female") {
                count = count + 1
        }
        Move X to Pile 2
}
```

Both code same

```
Fcount = 0
Mcount = 0
while (Pile 1 has more cards) {
        Read the top card X in Pile 1
        if (X.TownCity == "Chennai") {
                if (X.Gender == "Female") {
                        Fcount = Fcount + 1
                }
                else {
                        Mcount = Mcount + 1
                }
        }
        Move X to Pile 2
}
```

```
Fcount = 0
Mcount = 0
while (Pile 1 has more cards) {
        Read the top card X in Pile 1
        if (X.TownCity == "Chennai" AND X.Gender == "Female") {
                Fcount = Fcount + 1
        }
        else {
                Mcount = Mcount + 1
        }
        Move X to Pile 2
}
```

1st → Channai Male
2nd → not like 1st

PA : 3,8,9,11
GA : 6,7,8

Atleast → min → x >= value
Atmost → max → x <= value

Note :

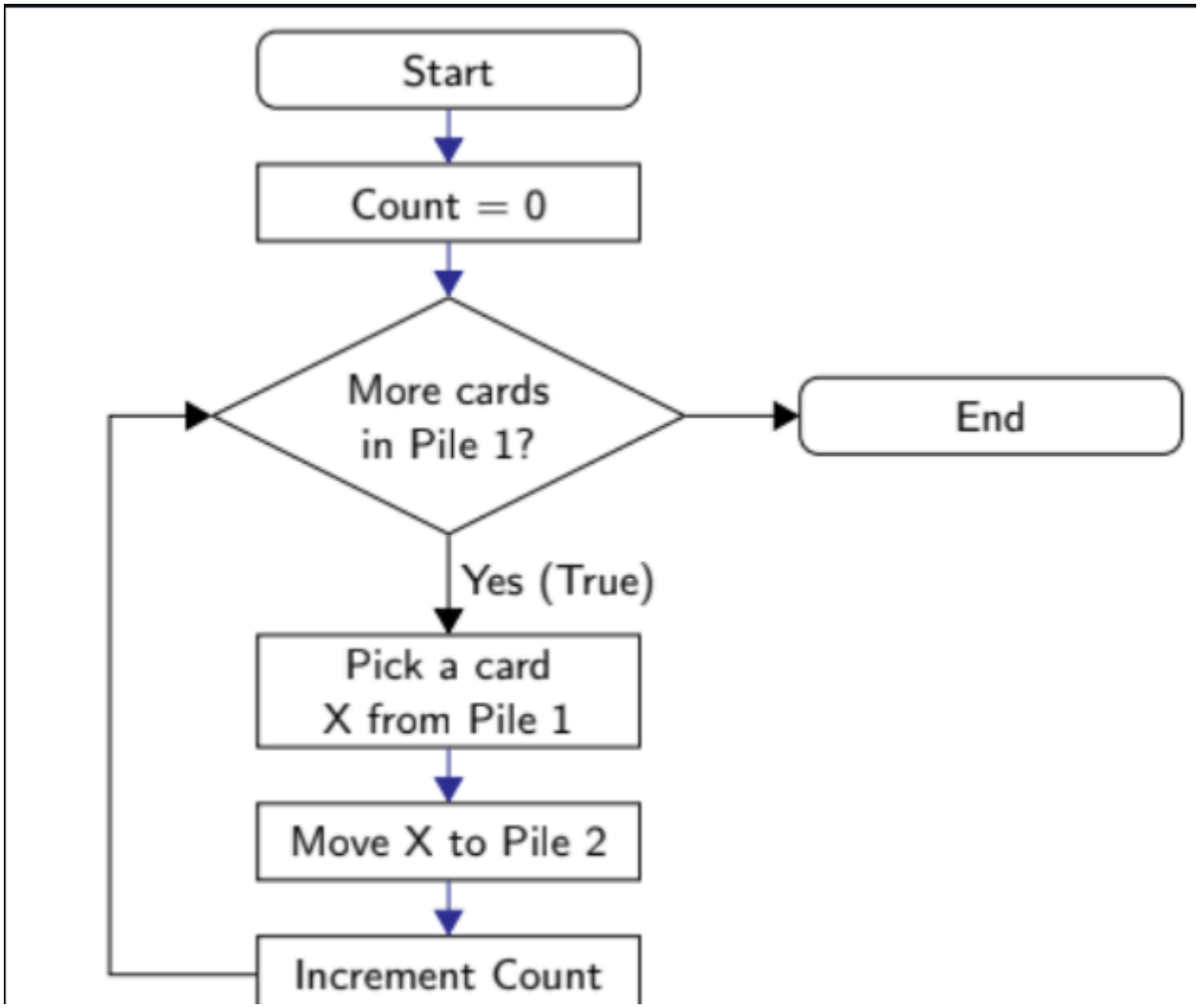
Week – 2

1. Flowcharts

What is a Flowchart?

A flowchart is a pictorial representation of a computational process. It shows the steps of a problem-solving process in a visual form using standard symbols.

Eg: Counting the number of cards



Node types:

1. Process (Indicated in rectangular box)
2. Decision (Indicated in diamond shape box)
3. Terminal (Used for start and end, indicated by oval)

→ **Arrows indicate operation flow**

Pros and cons of Flowcharts

Advantages:

- Visual representation of computation
- **Easy to understand**

Disadvantages:

- Size: Complex processes generate large flowcharts
- Collaboration: Sharing pictures in editable format
- Versions: Compare changes between flowcharts

2. From Flowchart to Stepwise Text

Instead of drawing, we can write the same process step by step in words:

Step 0 Start
Step 1 Initialize Count to 0
Step 2 Check cards in Pile 1
Step 3 If no more cards, go to step 8
Step 4 If yes, Pick a card X from Pile 1
Step 5 Move X to Pile 2
Step 6 Increment Count
Step 7 Go back to step 2
Step 8 End

Programming language

- Succinct notation for computational processes
- Better textual representation for
- conditional execution
- Step 3 If no more cards, go to Step 8
- Step 4 Pick a card X from Pile 1 (we want to write it somewhere in text that step depends on step 3 to be executed)
- AND Repeated execution
- Step 2 Check cards in Pile 1

.

.
.

- Step 7 Go back to step 2 (we also want to write that from step 2 to step 6 needs to be repeated again and again, which is not written in flow chart)

3. Pseudocode

What is pseudocode?

- A textual, language-like notation to describe computational processes.
- Easier to write and understand than flowcharts.
- Can be quickly converted into real code later.

```
Start
Count = 0
while (Pile 1 has more cards) {
    Pick a card X from Pile 1
    Move X to Pile 2
    Increment Count
}
End
```

1. Assign a value to a variable
2. Repeat steps while condition holds
3. Mark start and end of repeated block

Summary

- Flowcharts are easy to read, visual descriptions of procedures, but they are cumbersome; hard to share and edit.
- Writing down steps in a text is an alternative.
- Tune the notation to capture standard features:
    - Assigning values to variables
    - Conditional execution
    - Repeated execution

Pseudocode: Iteration and Filtering

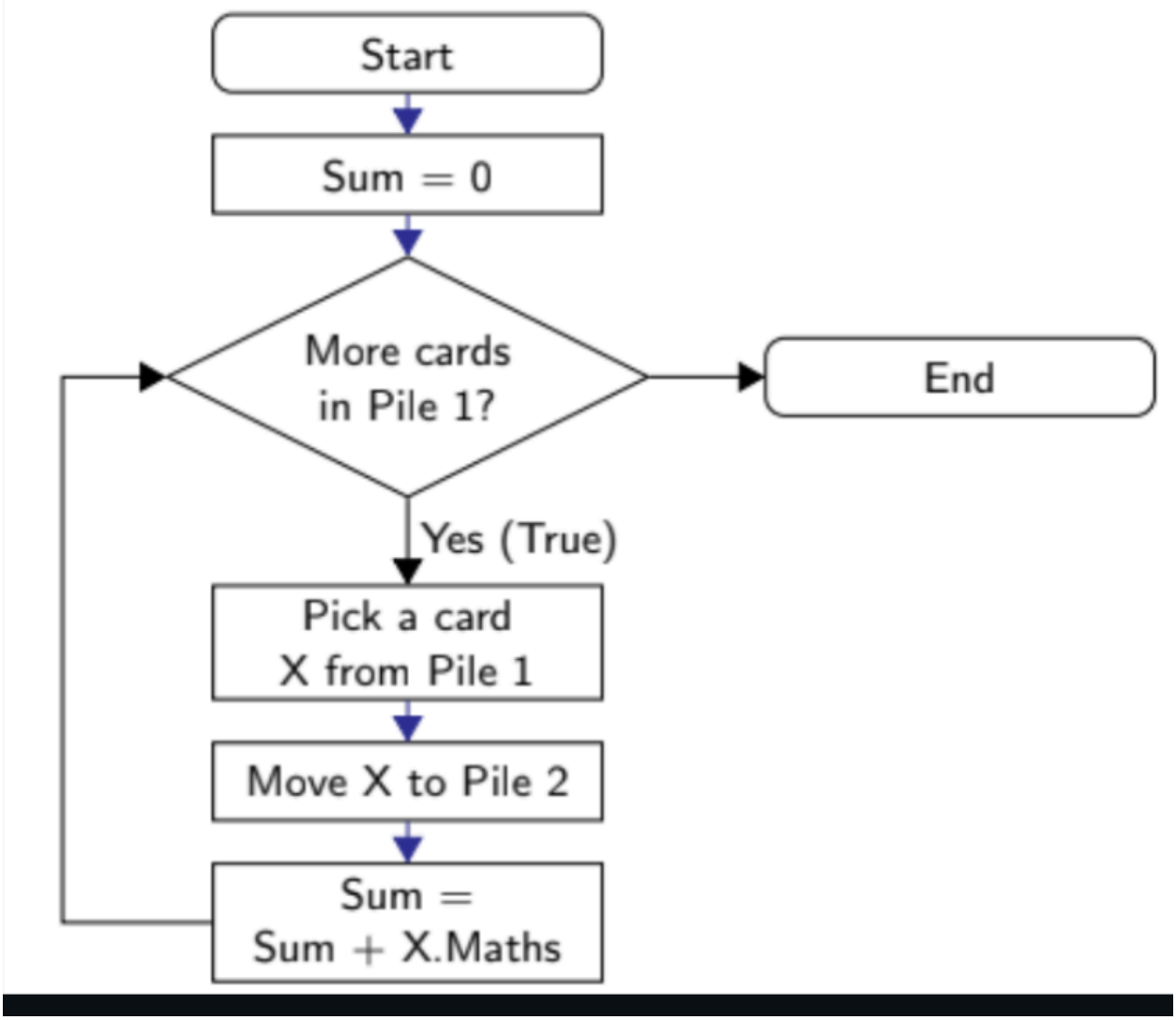
Counting cards

```
Start
Count = 0
while (Pile 1 has more cards) {
    Pick a card X from Pile 1
    Move X to Pile 2
    Increment Count
}
End
```

Will dispense with Start and End henceforth
And now let's modify the pseudocode to get the sum of maths marks…

Sum of Maths marks

```
Sum = 0
while (Pile 1 has more cards) {
    Pick a card X from Pile 1
    Move X to Pile 2
    Sum = Sum + X.Maths
}
```



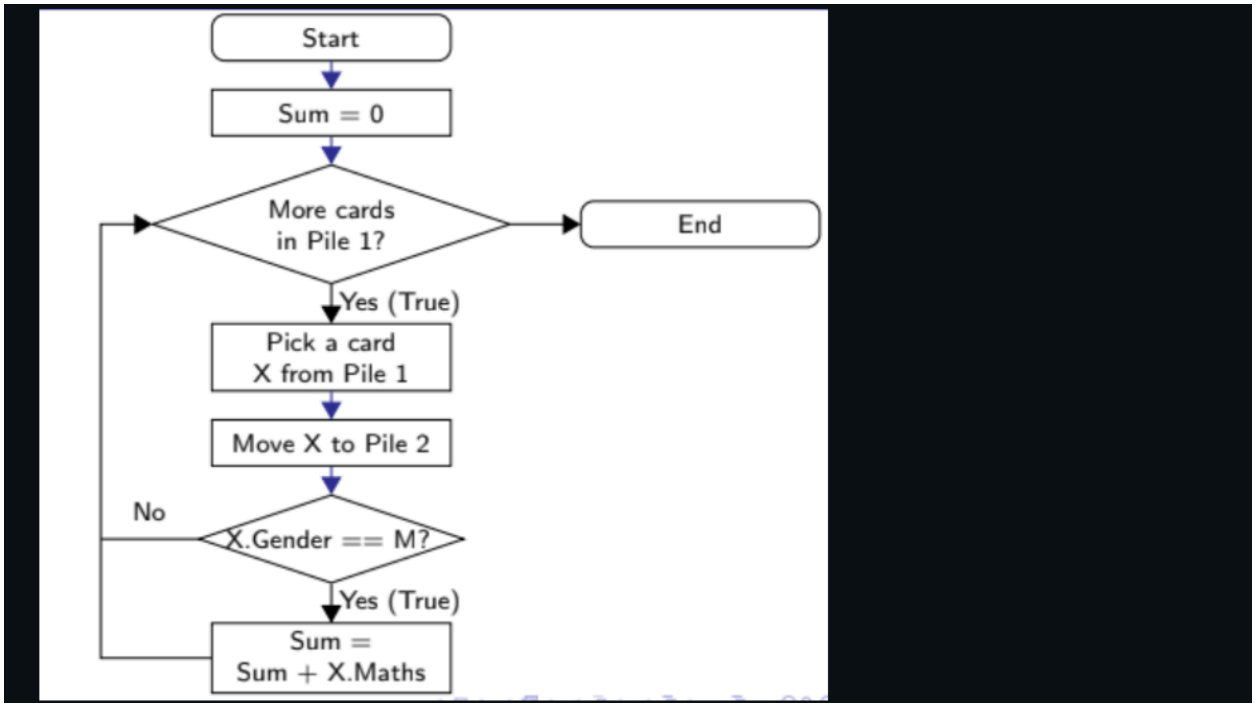
- **Update Sum: assignment statement**
  - Sum on right is current value
  - Sum on left is updated value
  - = is not mathematical equality

- **Increment: Count = Count + 1**
  - x.Maths: Maths marks in card x

Sum of Boys' Maths marks

```
Sum = 0
while (Pile 1 has more cards) {
    Pick a card x from Pile 1
    Move x to Pile 2
    if (x.Gender == M) {
        Sum = Sum + x.Maths
    }
}
```

Flowchart:

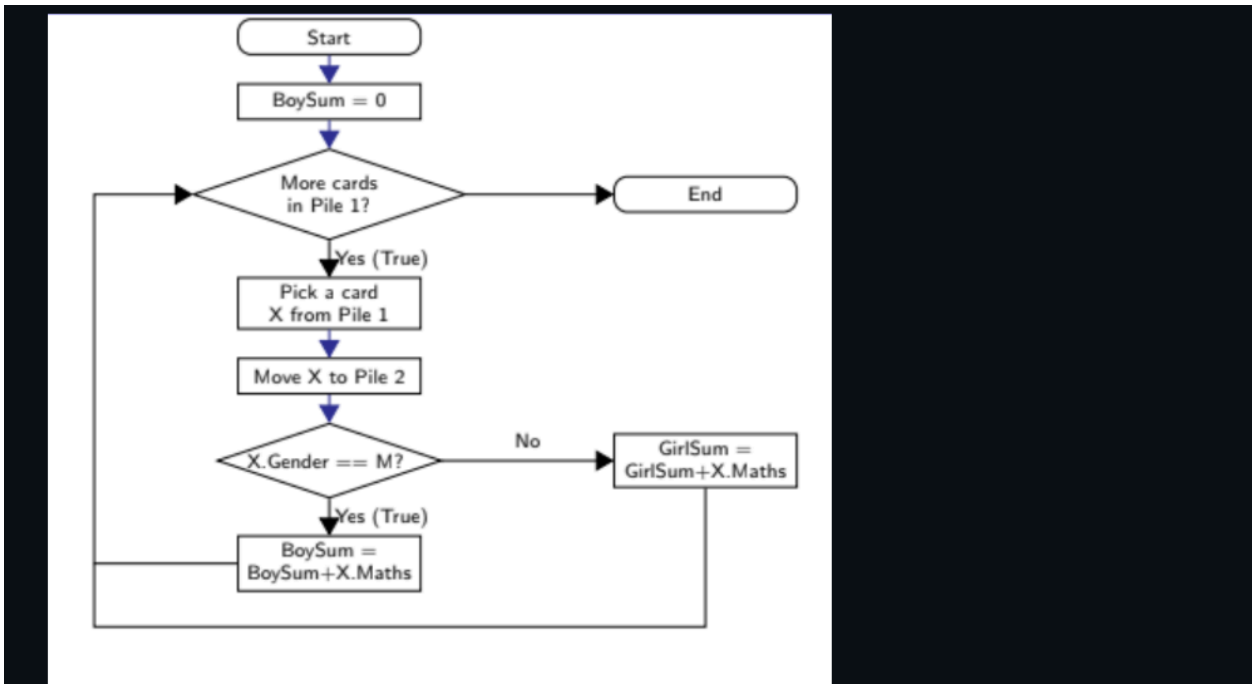


- **Conditional execution error**
- **Equality (==) vs Assignment (=)**

Sum of Boys' and Girls' Maths marks

```
BoySum = 0
GirlSum = 0
while (Pile 1 has more cards) {
    Pick a card x
    Move x to Pile 2
    if (x.Gender == M) {
        BoySum = BoySum + x.Maths
    } else {
        GirlSum = GirlSum + x.Maths
    }
}
```

Flowchart:



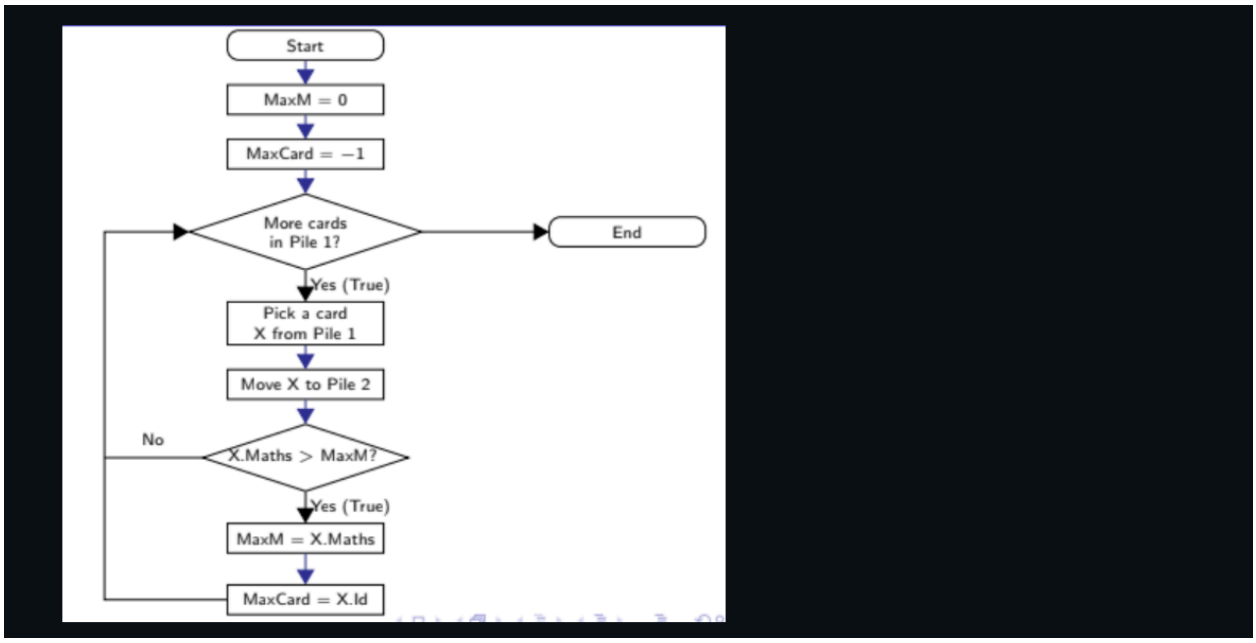
- **Alternative branch for conditional**

Finding the card with maximum Maths marks

```
MaxM = 0
MaxCard = -1
```

```
while (Pile 1 has more cards) {
    Pick a card x from Pile 1
    Move x to Pile 2
    if (x.Maths > MaxM) {
        MaxM = x.Maths
        MaxCard = x.Id
    }
}
```

Flowchart:



Summary

- **Assignment statement**
  - Count = 0
  - Sum = Sum + x.Maths
- **Conditional execution**
  - Once
    - if (condition) { ... }
    - if (condition) { ... } else { ... }
  - Repeatedly
    - while (condition) { ... }
- **Equality (==) vs Assignment (=)**
  - if (x.Gender == M)

4. Why Pseudocode?

- Flowcharts → good for visualization, especially for beginners.
- But pseudocode → easier to write, edit, and share.
- It looks close to real programming languages (like Python, C, Java).
- Natural next step after learning flowcharts.

5. Final Takeaway

- Start with flowcharts to visualize processes.
- Move to stepwise text for clarity.
- Then use pseudocode for precision and ease of conversion into code.
- Eventually, pseudocode transitions smoothly into real programming.

This structure makes the learning journey very logical:

Flowchart → Text Steps → Pseudocode → Programming.