

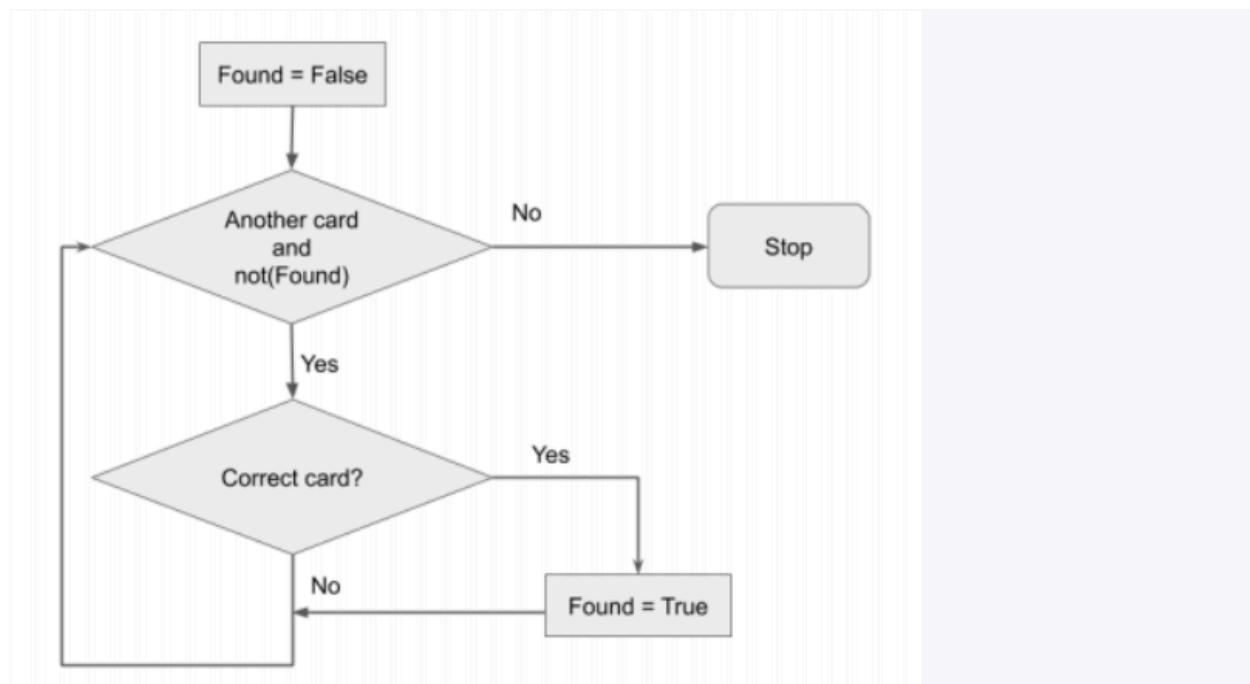
Week 2 : Iteration , Filtering , Selection , Pseudocode , Finding max and min , AND operator

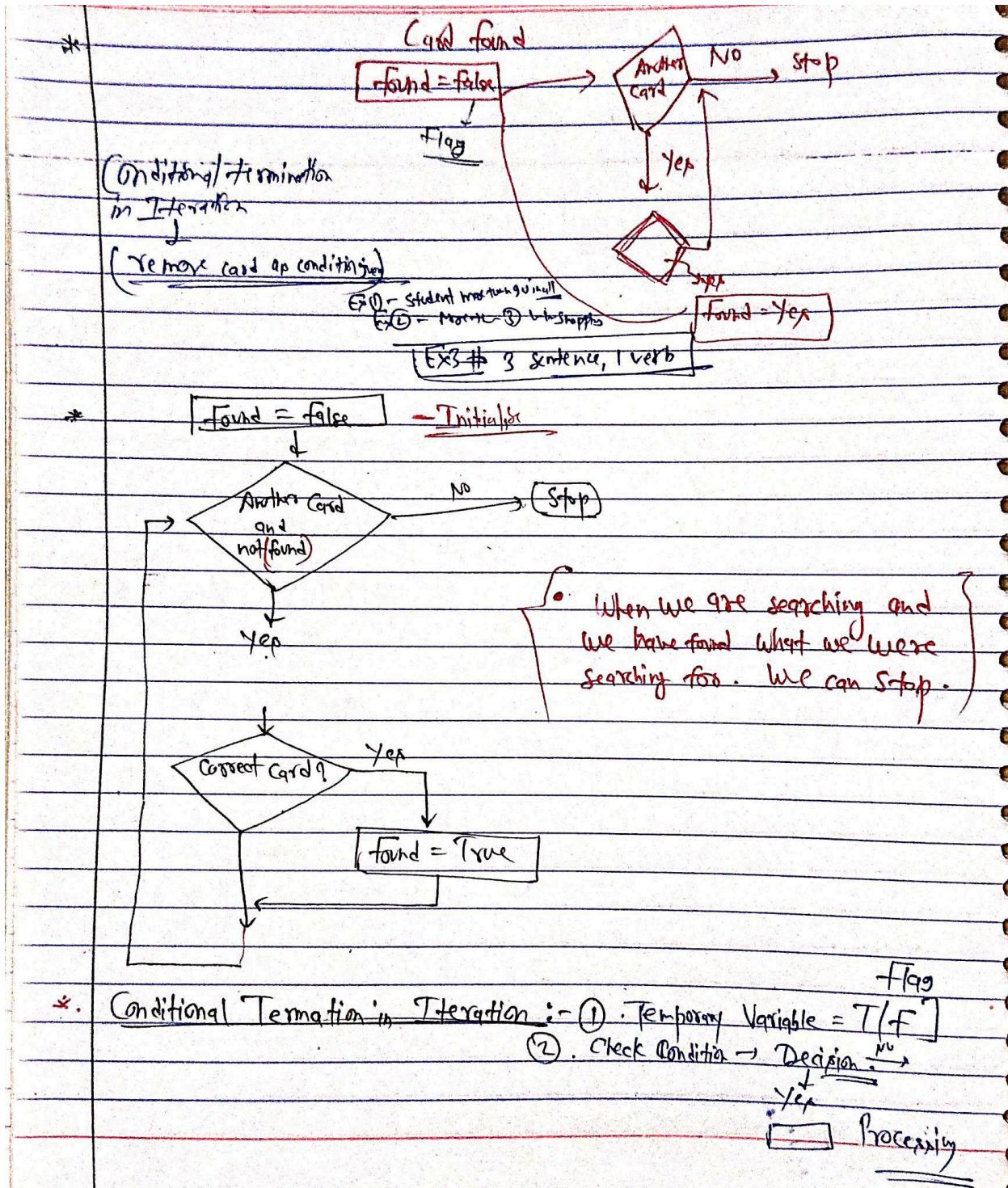
Lecture 1 : Conditional termination in Iteration

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

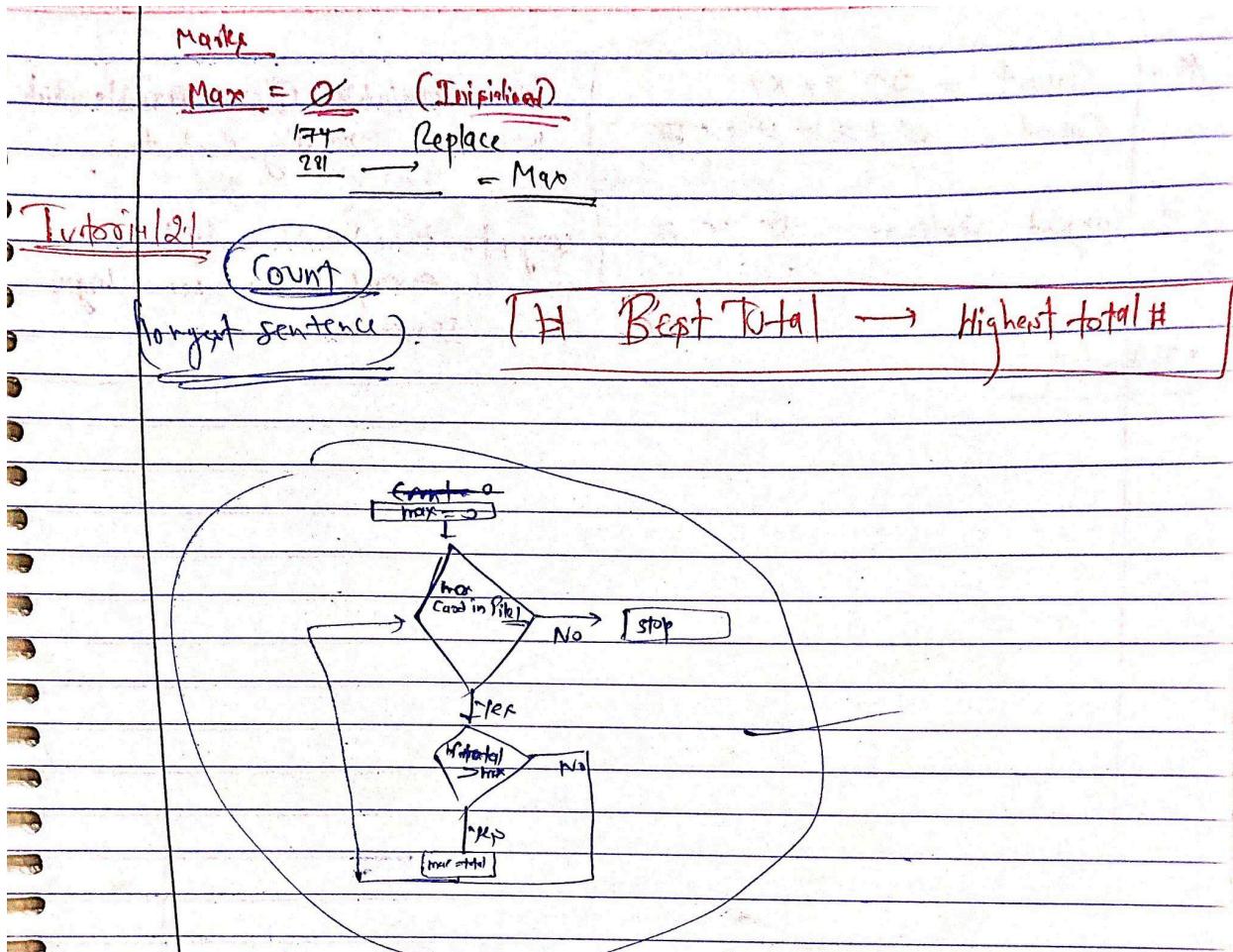
Definition

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





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)

*	Count = 0 1 2 3 4 Count = 0 1 2 3 4 5 6 7	Two Variable Count Variable which we keep restarting each time.
•	longest sentence = 0 4 7.	longestsentence Variable - Update every time we see a longer sentence.
Tutorial - 2.2 in		# Check if sentence word count # check length sentence (Longest sentence)
Ex	# Same month of birth then check which month highest max birth in a month =	
	Count - changes when month changes max birth in month - keeps track birth is maximum.	

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

* Shopping Bill :-

Shop → SV Count
SG Count

Count

SV = {1, 2, 3} → 15
BV = {1, 2} → 6
SG = {1, 1, 1} → 9

Max

which one is max

Shop → bill count for every shop
In each shop → bill count max at 3rd value on max
keep max = 0 and iter single iteration if max = Count (max / sv, bv, sg)

Tutorial Q.3 :- # City with max numbers of students
→ keep city count for different city and max diff. city = 0
→ Then check which city have most students.

QN : 2,3

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

	$\text{Shopping Bill} \leftarrow \frac{\text{Total Values}}{\text{Money instead of Bill}}$
# Check for Max	<u>Maxim Check</u> # Check money instead of Bill
	# Check shop
	# Then Add their Total Bill price
	# Check which one has max total.
<u>Tutorial Day</u>	<u>SV, VB, SC</u> → <u>Max</u>
#	• Which part of speech contributes most characters to the text? calculate letters count.
#	Check each part of speech letter count, and find which one is max.
	$\text{pronounlettercount} = 0$ $\text{verb} = 0$ $\text{AT} = 0$
	→ Check type of word and # letters count to respective variable Compare to max, and maximum variable is max.

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-	
morning	1	the	6	work	1	nayagam	1
Swaminathan	1	calendar	1	discipline	1	class-	
reluctant	1	After	1	shuddered	1	teacher	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-	
morning	1	the	6	work	1	nayagam	1
Swaminathan	1	calendar	1	discipline	1	class-	
reluctant	1	After	1	shuddered	1	teacher	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		

Q1 Word Counter

Revise

Max frequency - 6

IT	2
Was	3
Monday	3
Morning	1

The 6

→ Word frequency [Ans]

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

Iteration 2.5 :- i - No of bills per customers,

→ Declare Variable for no of customers } - 1st Iteration -

→ Increment after each Iteration } -

→ In last check which one variable has max value.

→ maxbill (One Iteration)

QN 1,2

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

max mark find up before, but this time also need id for that card.

Max Mark - ? # check mark mark and then update max card no.

Max card no -

ex) Max Mark marks and card no

① Work max occurs once.

→ check which occur more than one
→ remove from set.

Tutorial 2.6 : - # Max bill amount and that customer ID ?

Max bill amount = 0.

max bill customer : string (None) :- None

Tutorial 2.7 : - # Initialisation of variable

count = 0 , sum = 0 , Name = None,

After Iteration, update val.

Min , Max

33 43 93 50 All others falling in this Range.
X min

Range of data, 0 — 100
Initialise → Max — Min

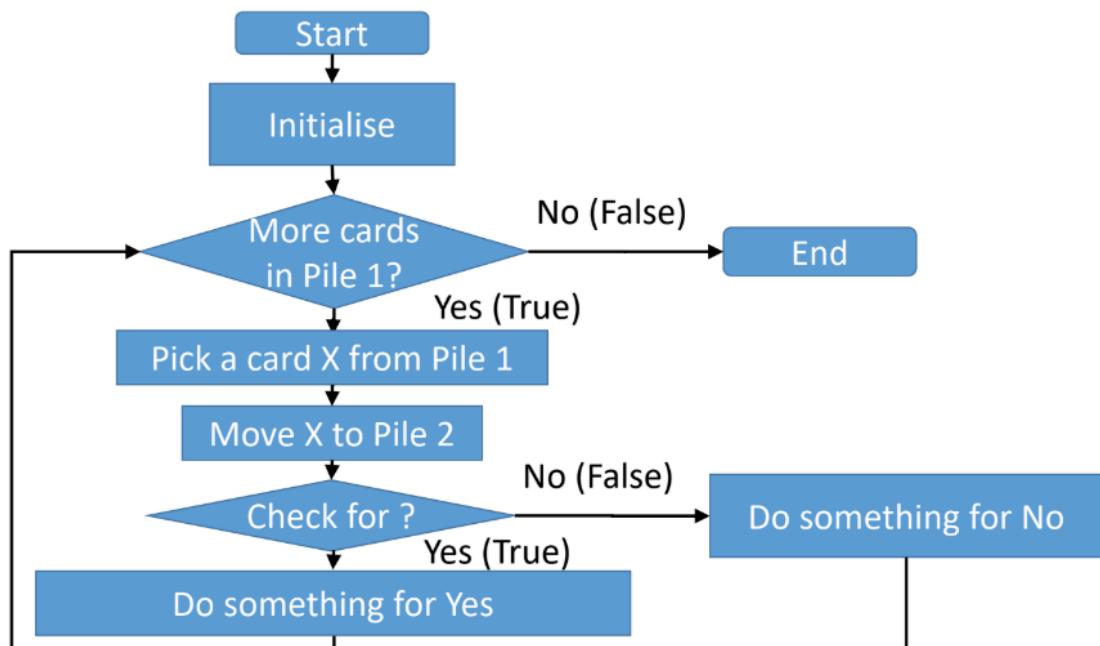
No range fix, min = 100000 , max = -1

QN : 3

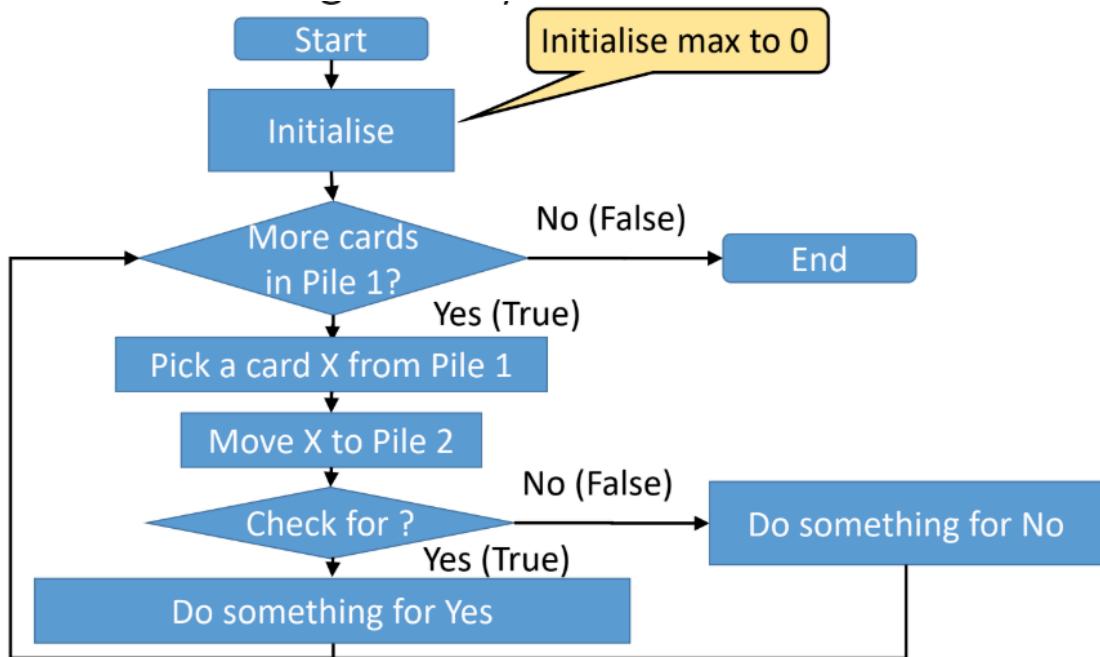
Lecture 8 : Flowchart for max marks

Flowchart for Max of Maths

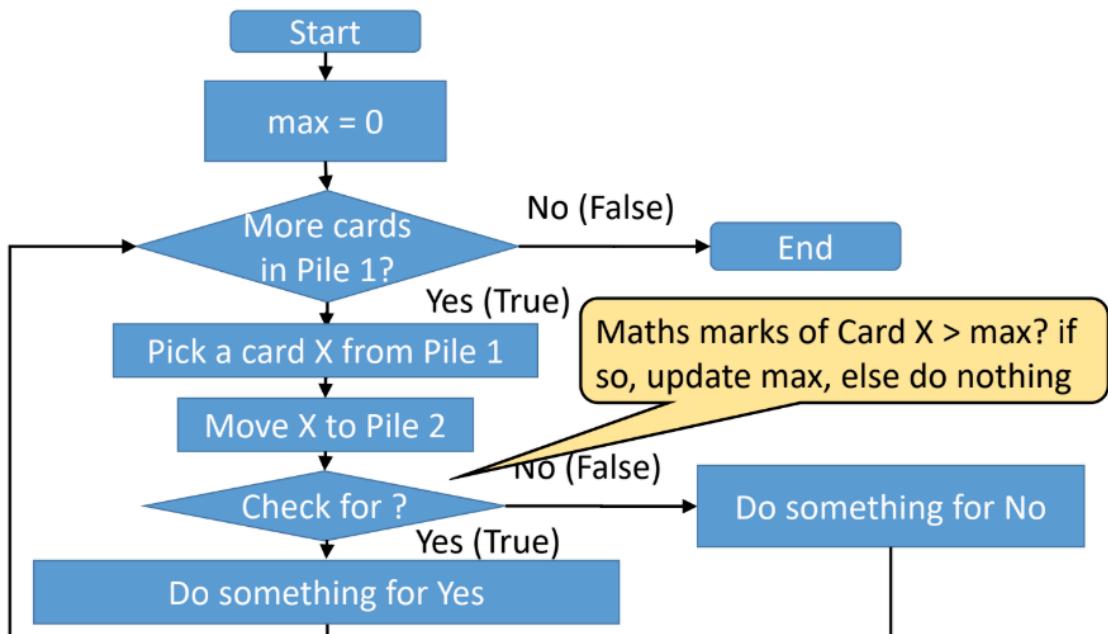
Iteration with filtering



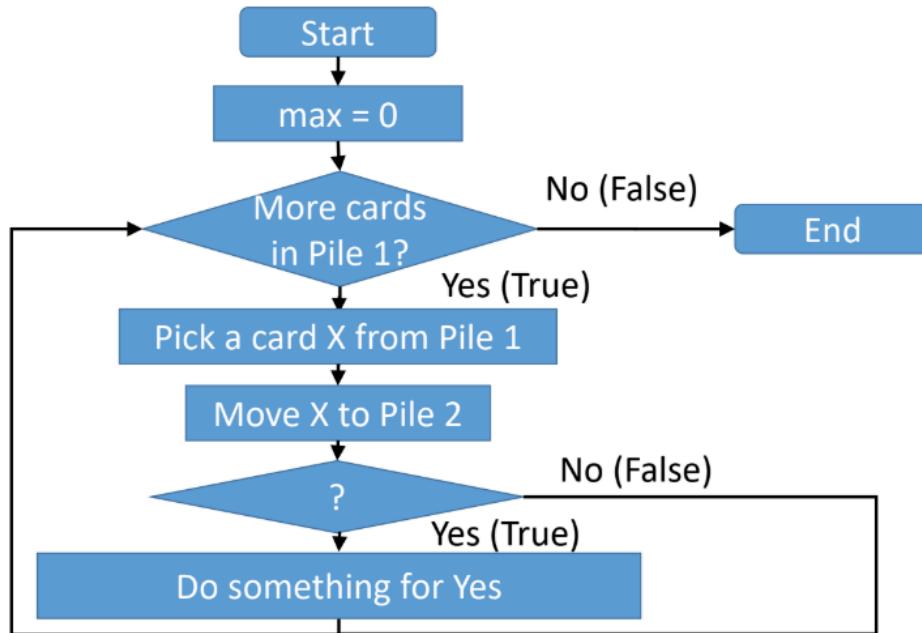
Iteration with filtering: modify for max



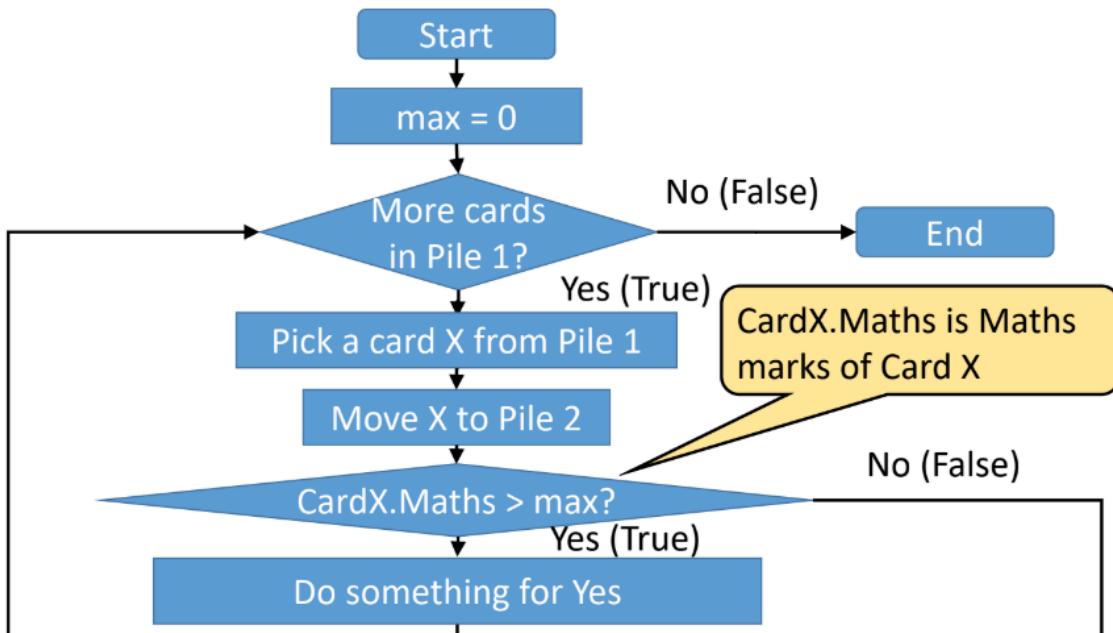
2.



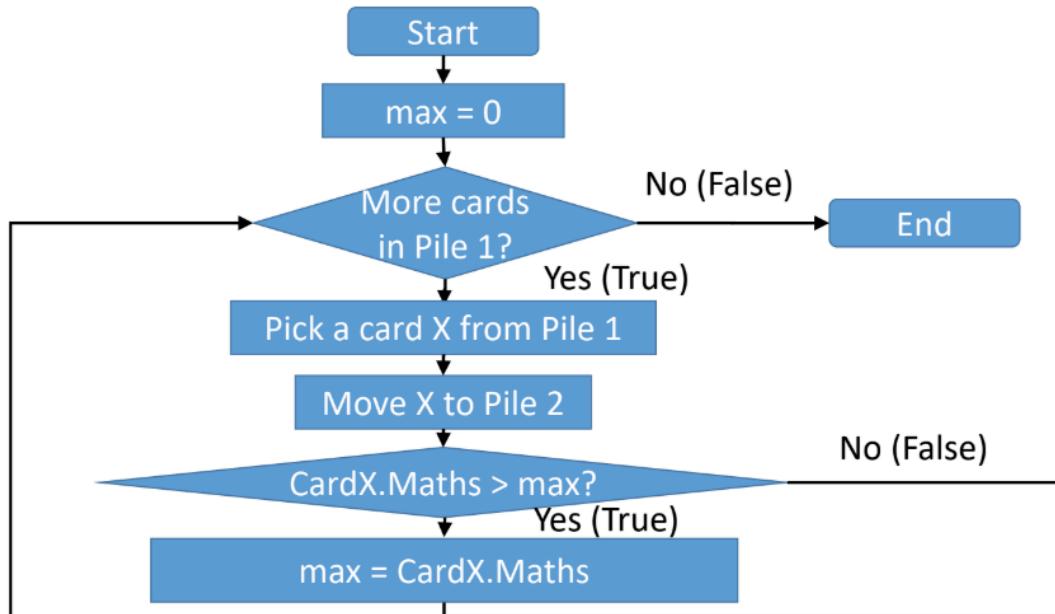
3.



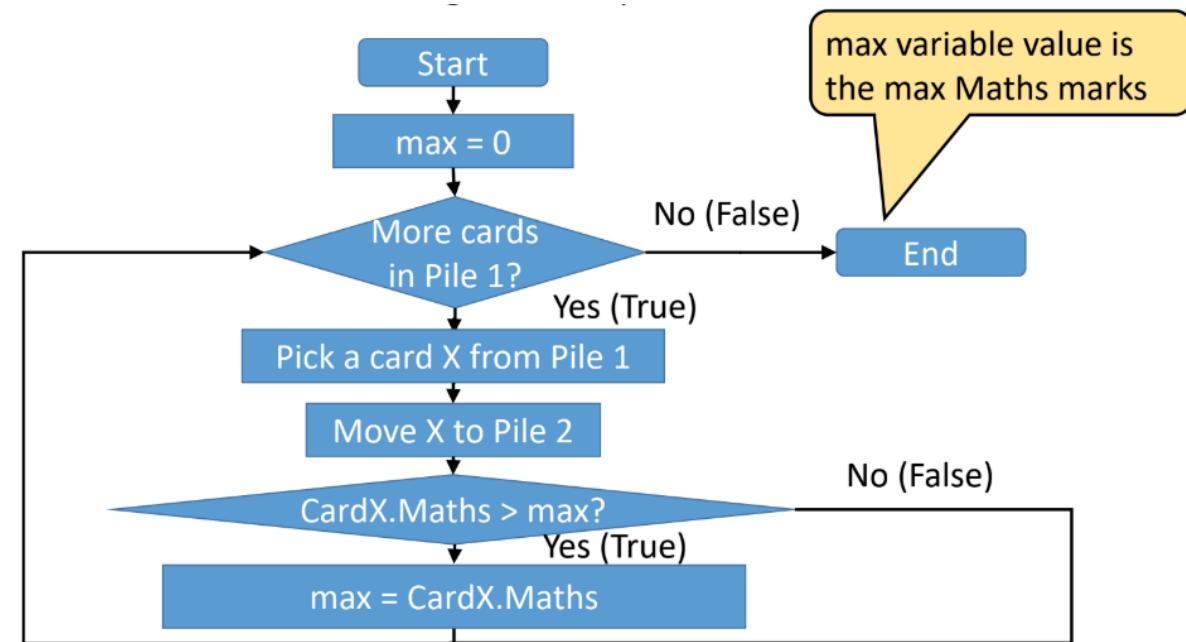
4.



5.

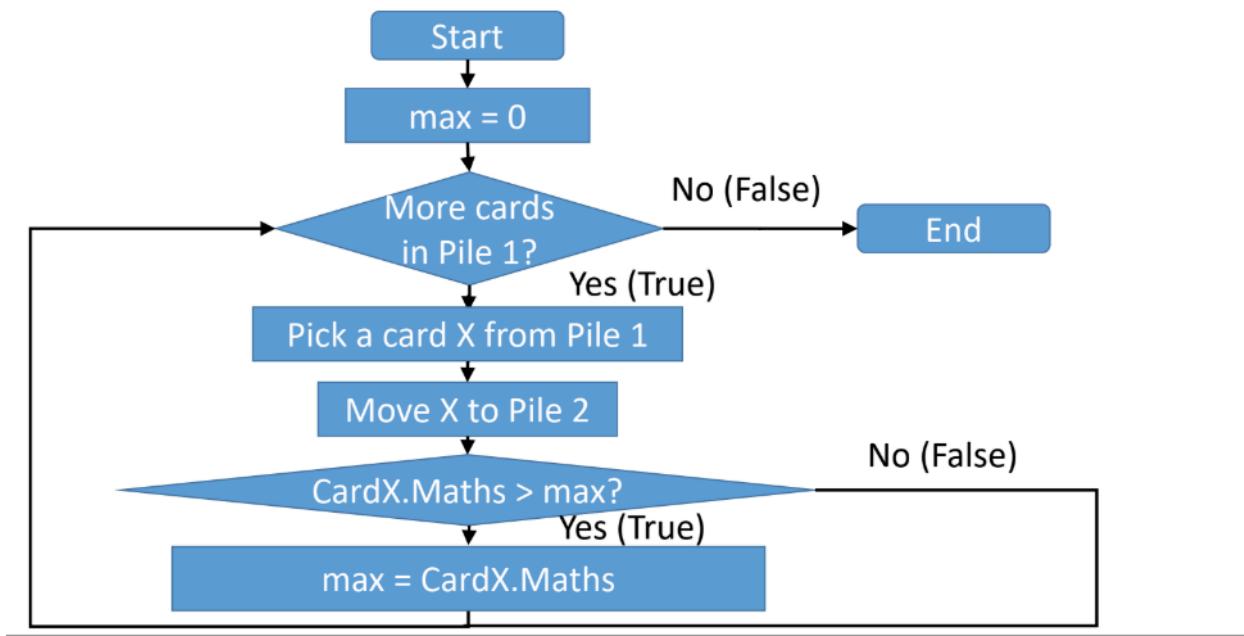


6.

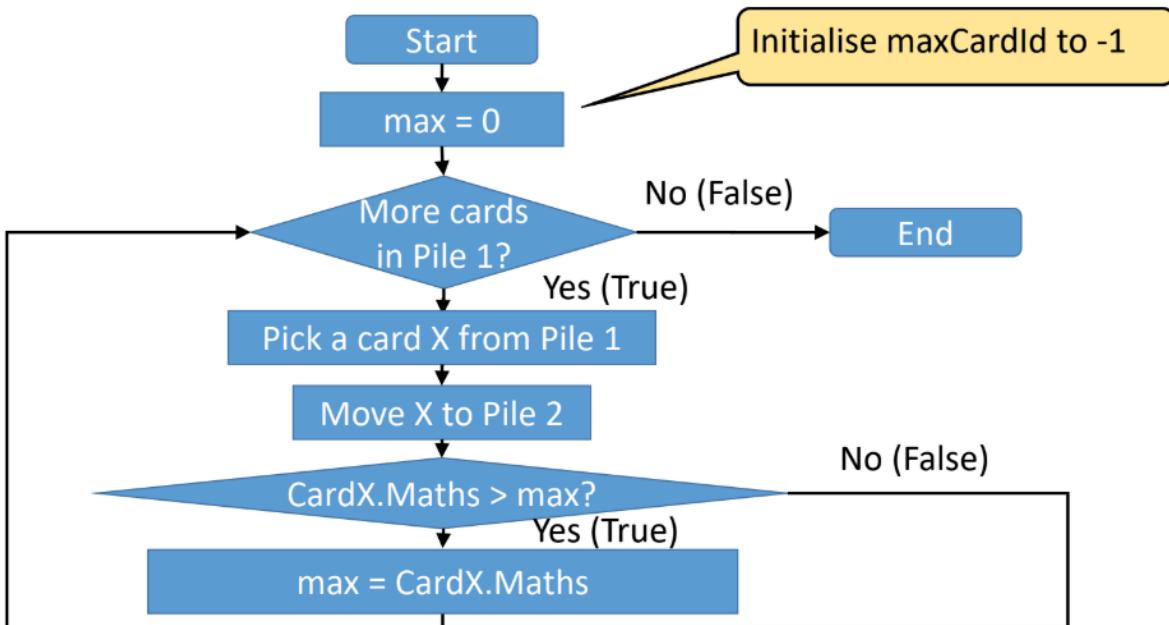


Flowchart for Max of Maths Marks; keep track of the card which has the max marks

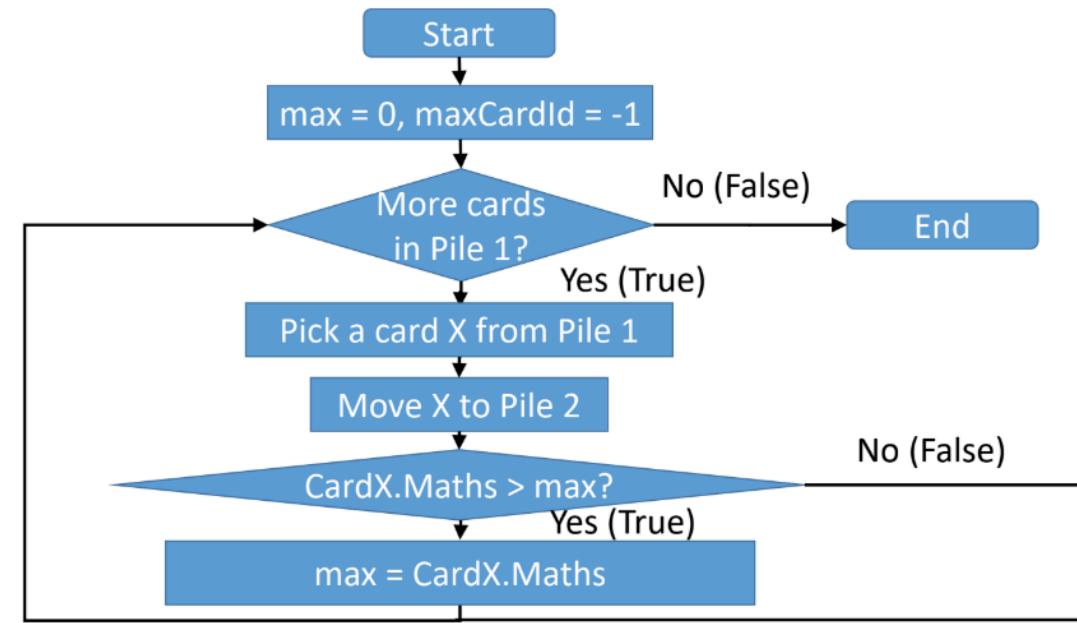
Max of Maths marks: keep track of card



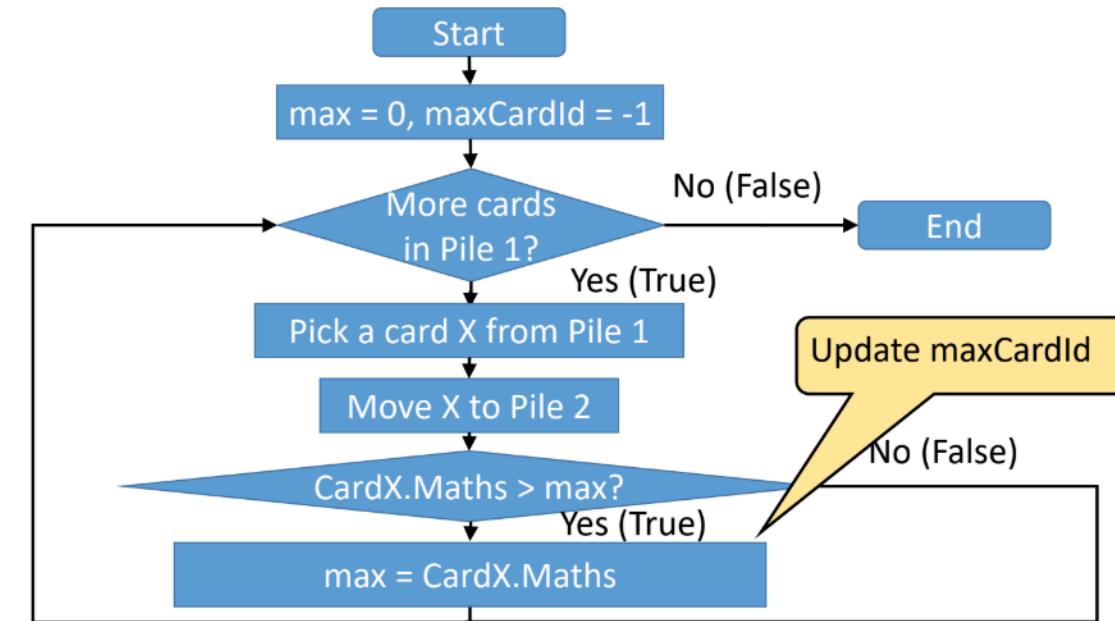
2.



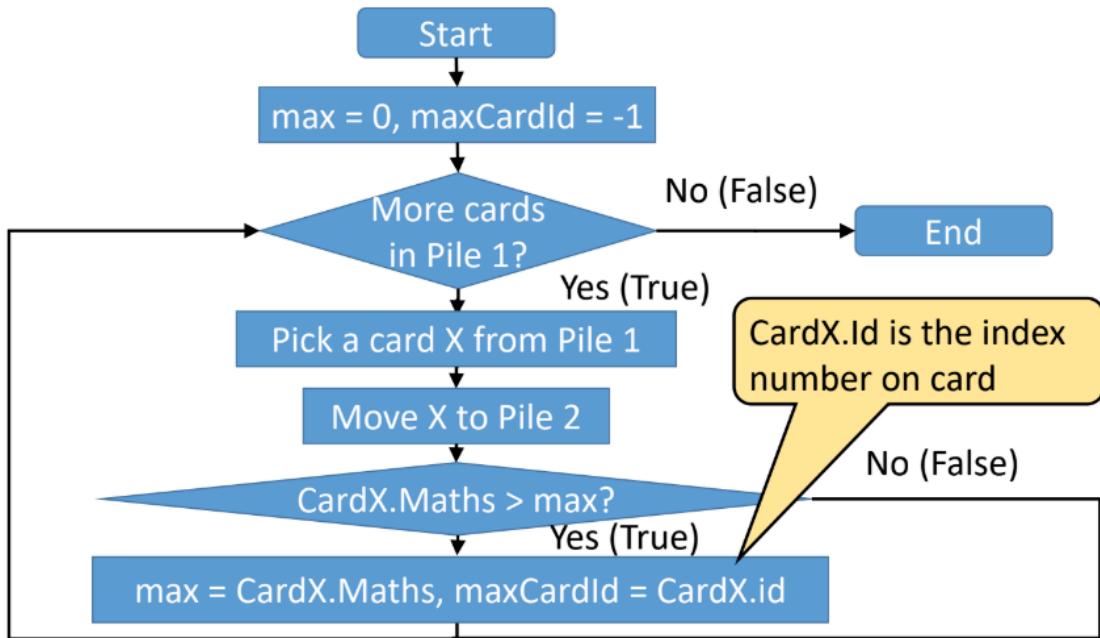
3.



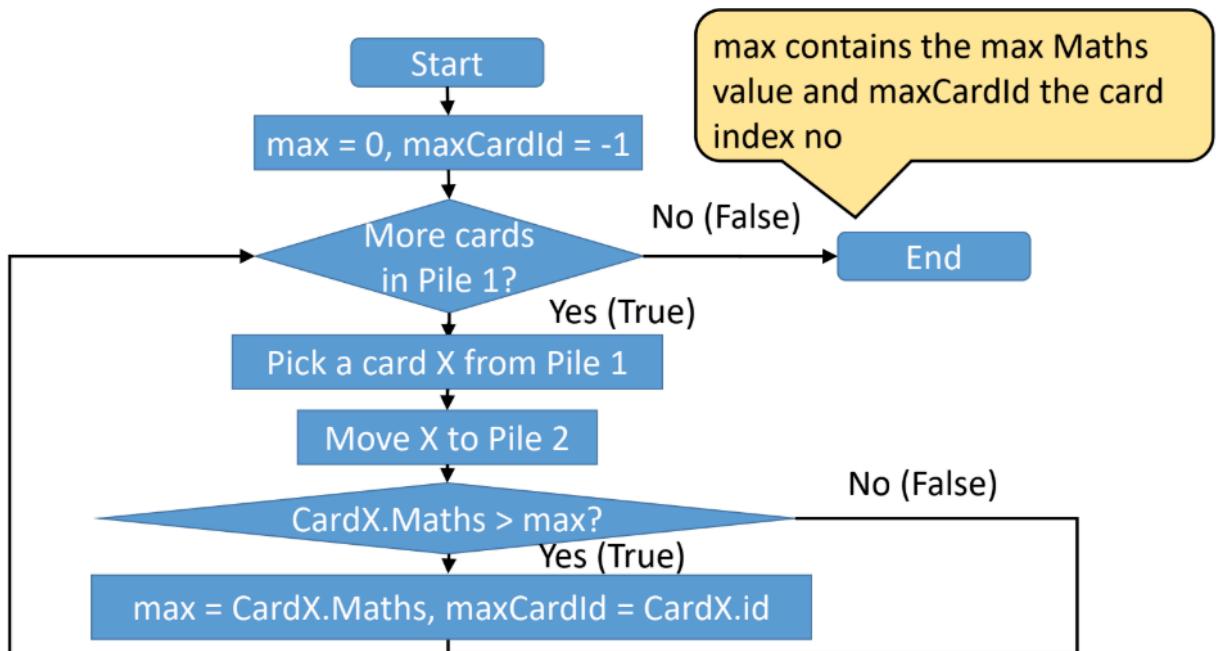
4.



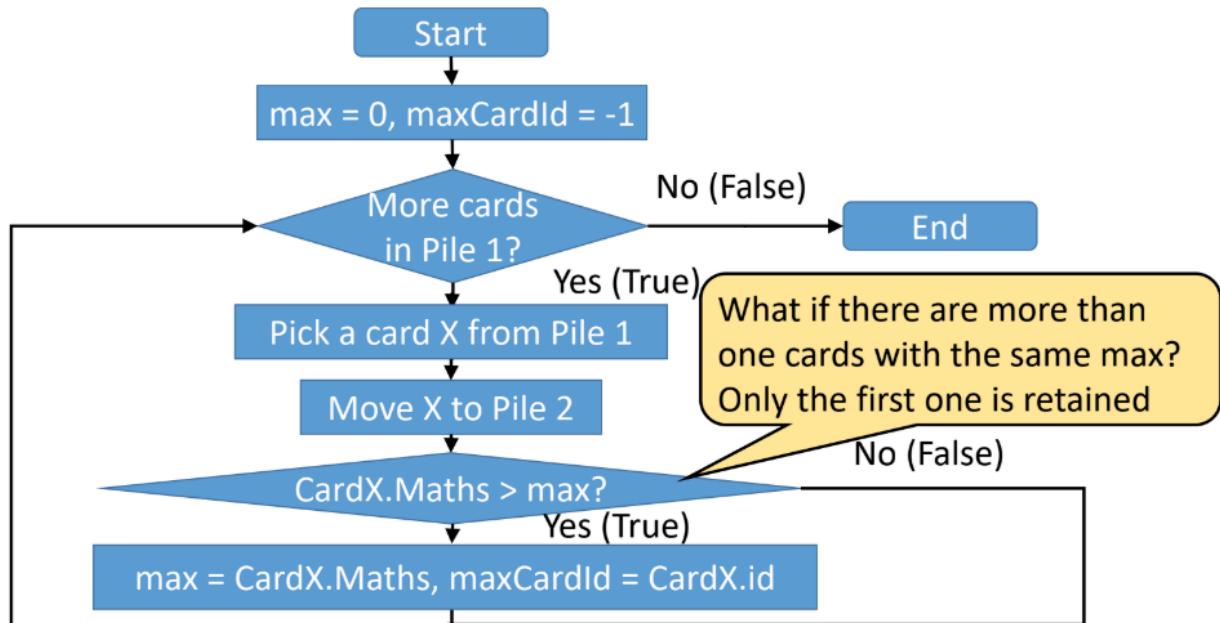
5.



6.

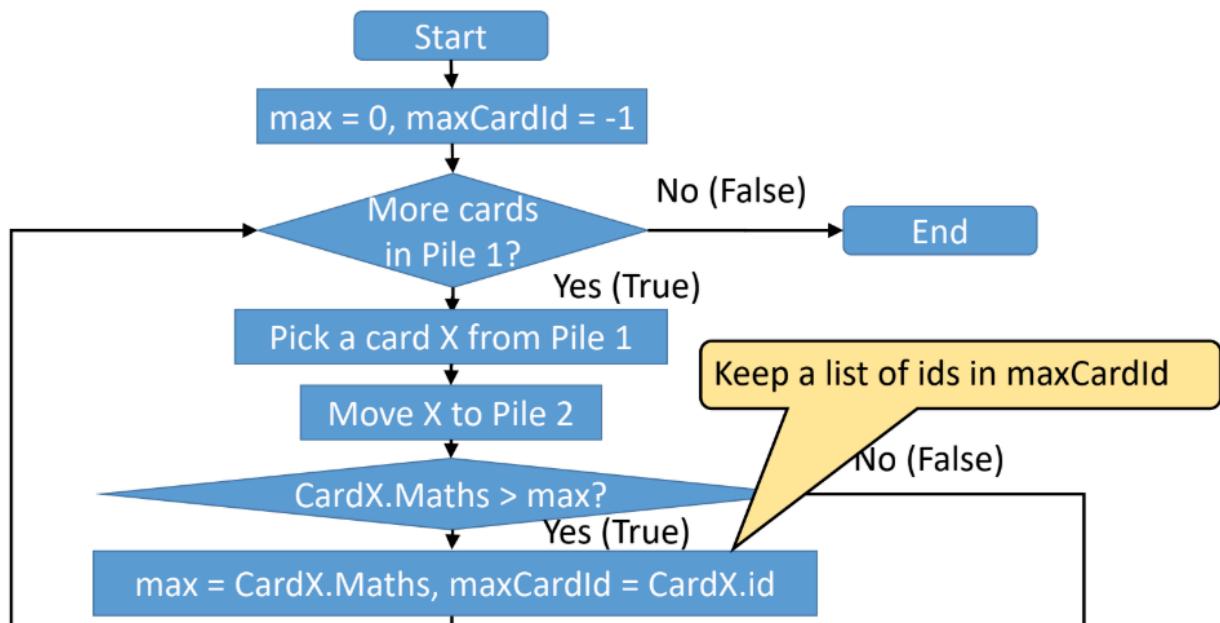


Max of Maths marks: keep track of card - PROBLEM !

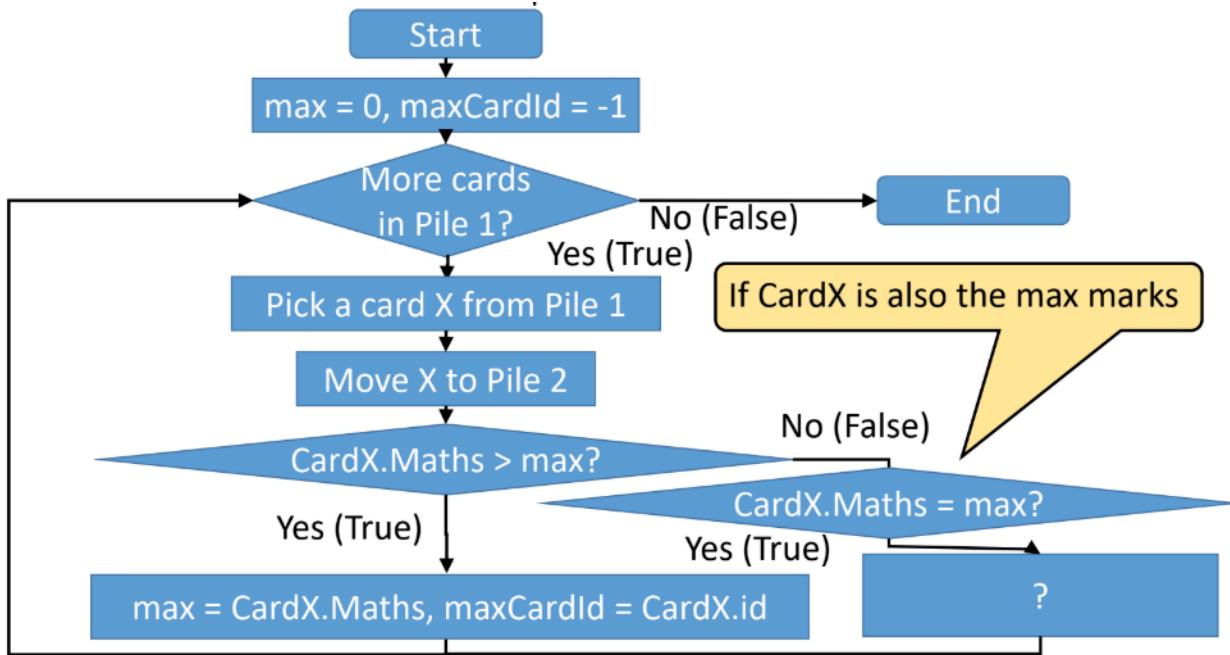


Max of Maths marks: keep track of card - SOLUTION !

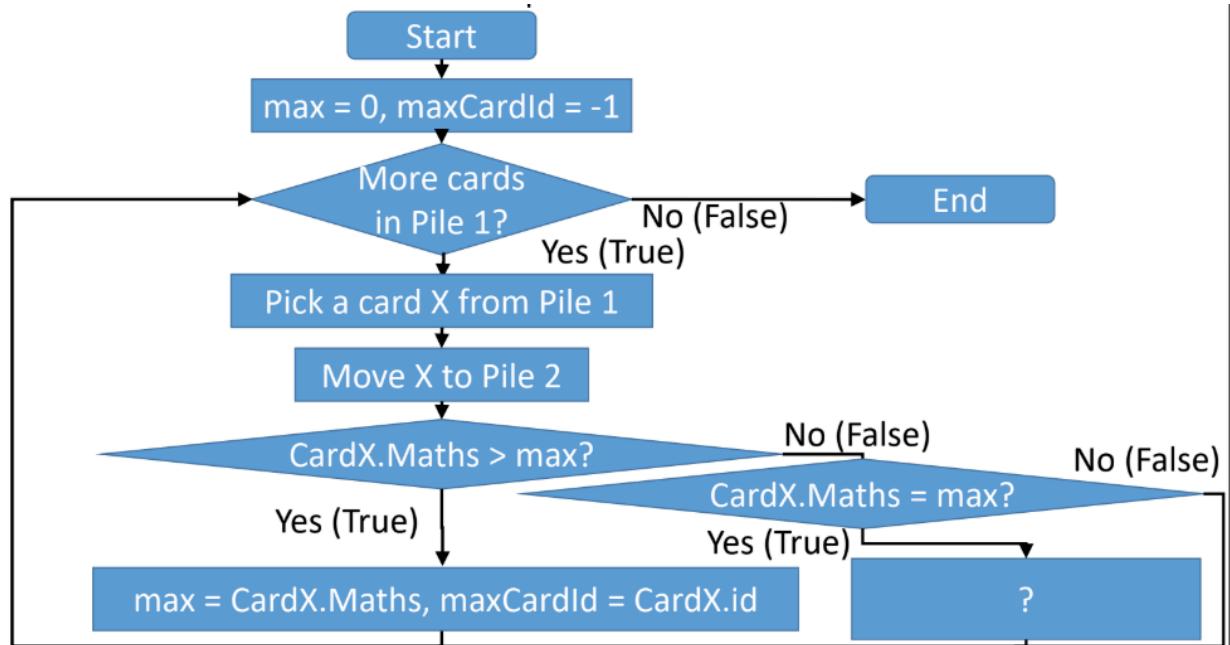
1.



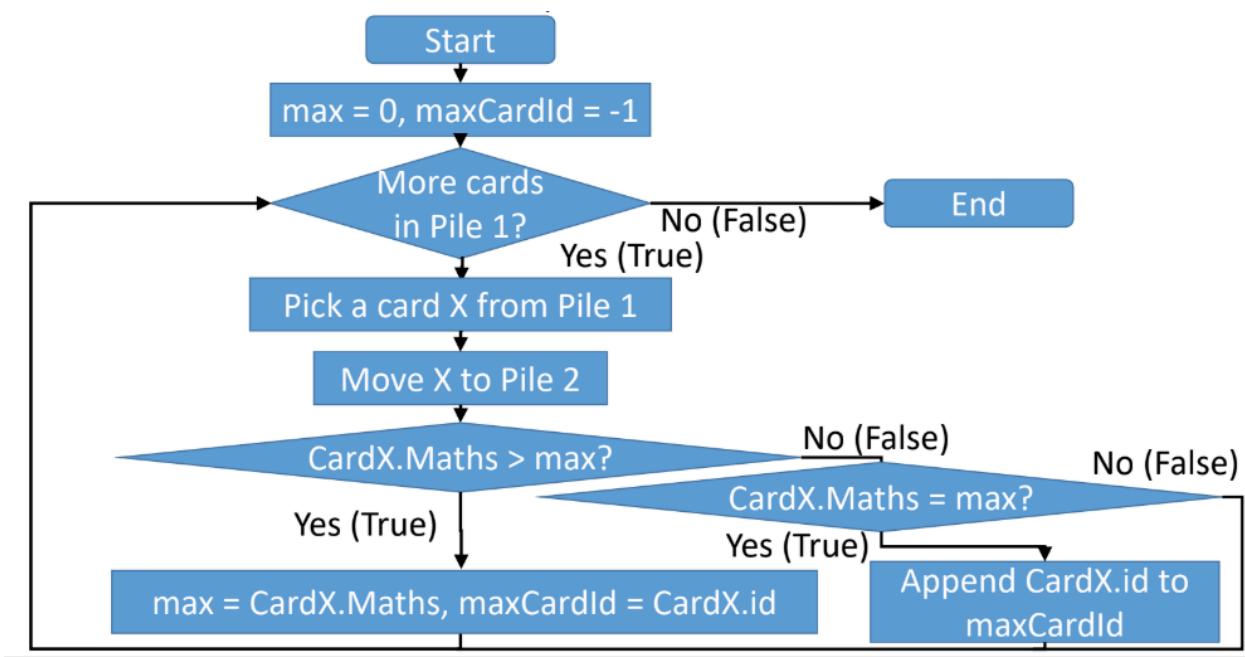
2.



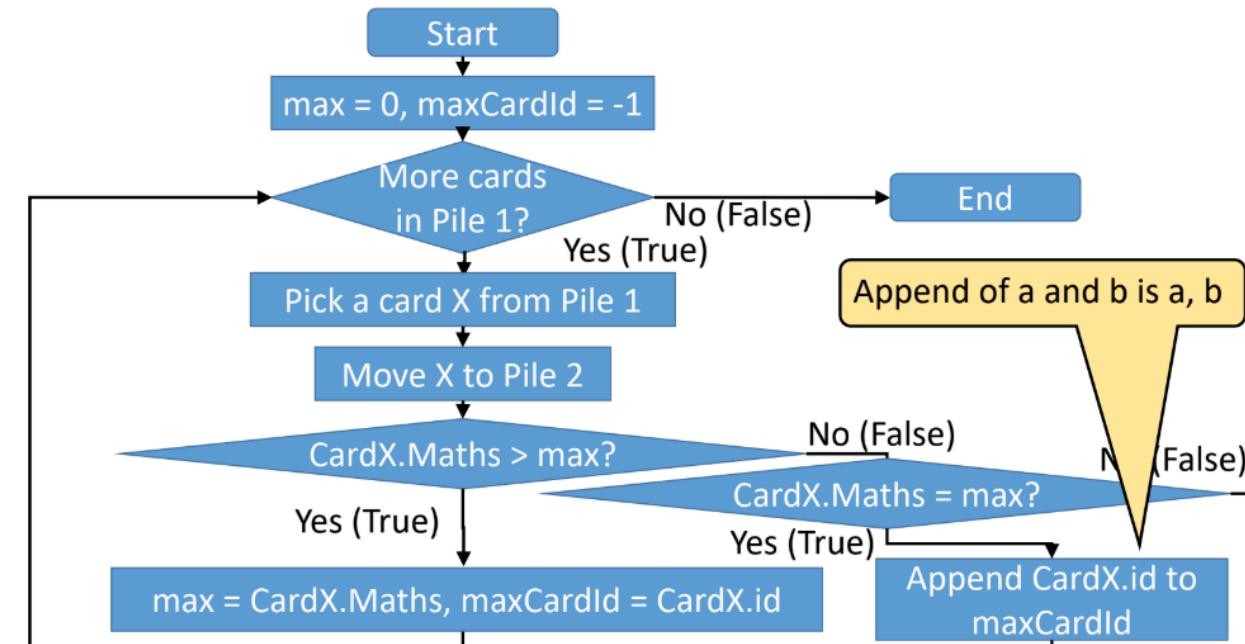
3.



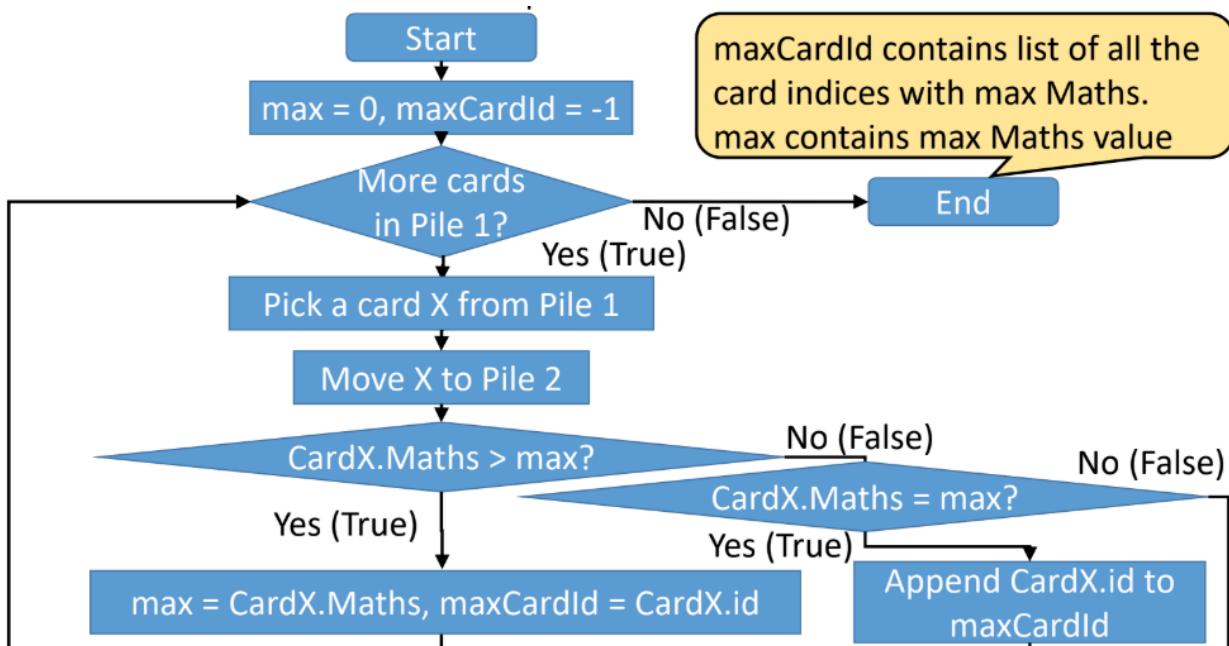
4.



5.



6.



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.).

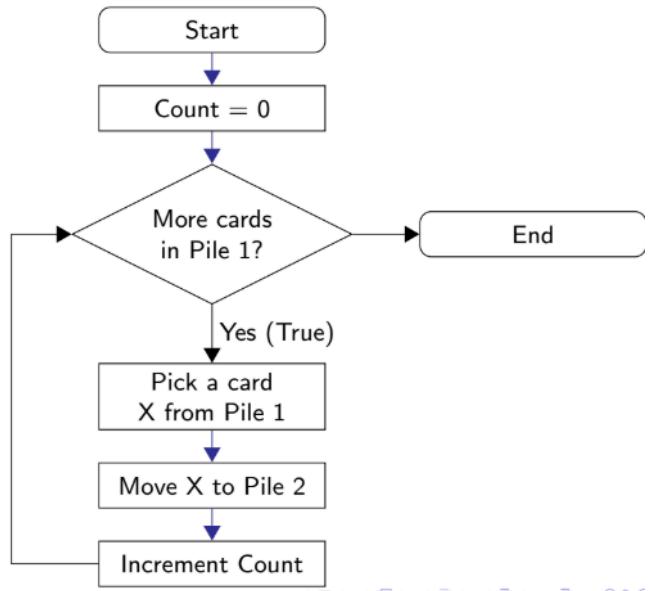
It helps programmers plan before writing real code.

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



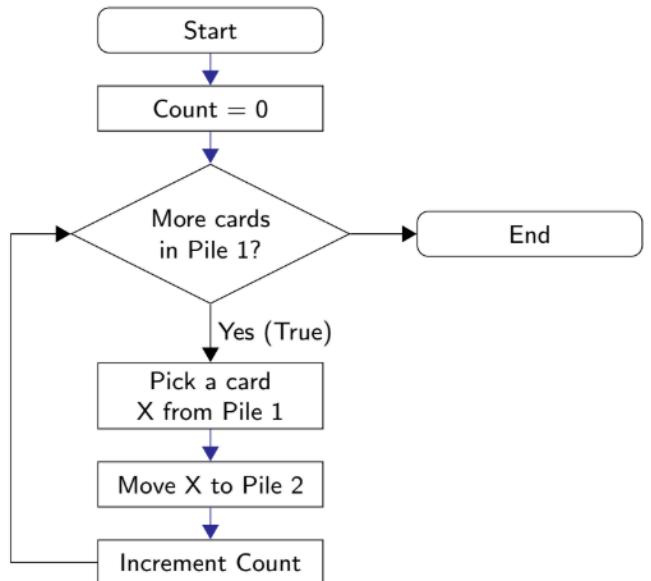
Pros and cons of flowcharts

Advantages

- Visual representation of computation
- Easy to understand

Disadvantages

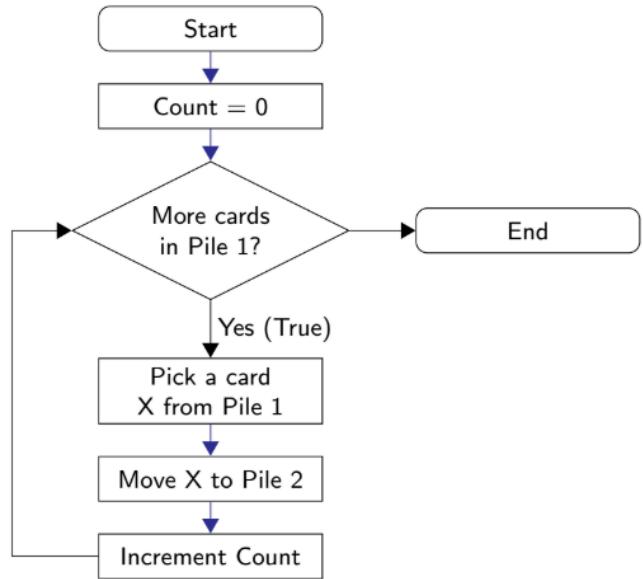
- Size: Complex processes generate large flowcharts
- Collaboration: Sharing pictures in editable format
- Versions: Compare changes between flowcharts



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



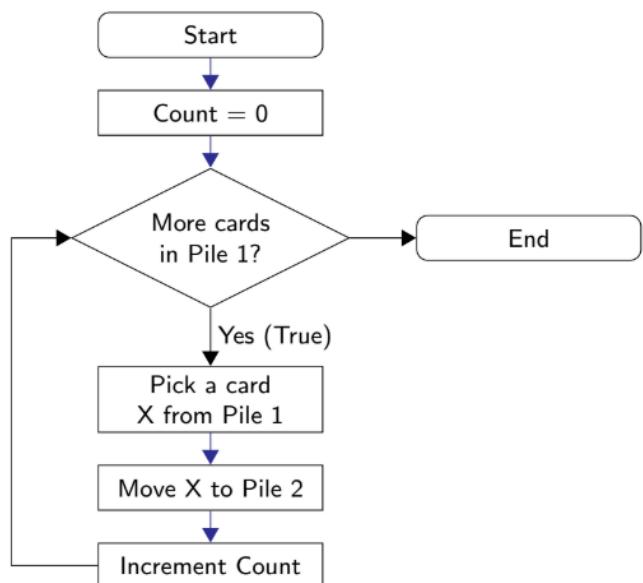
Programming language

- 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

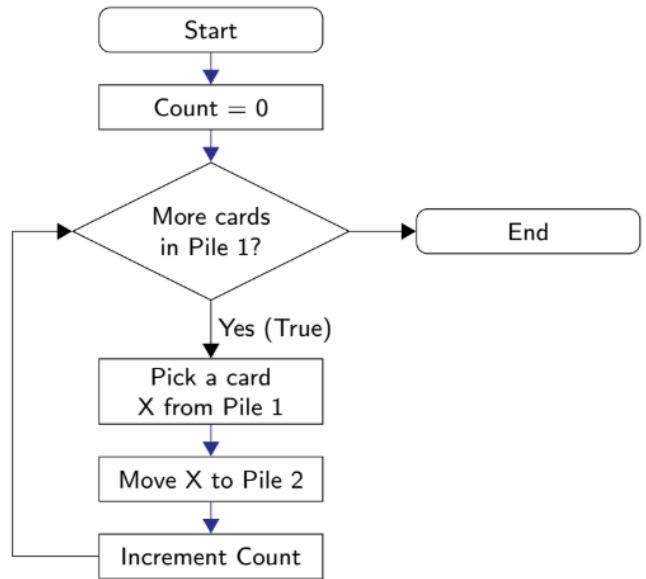
Repeated execution

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
```

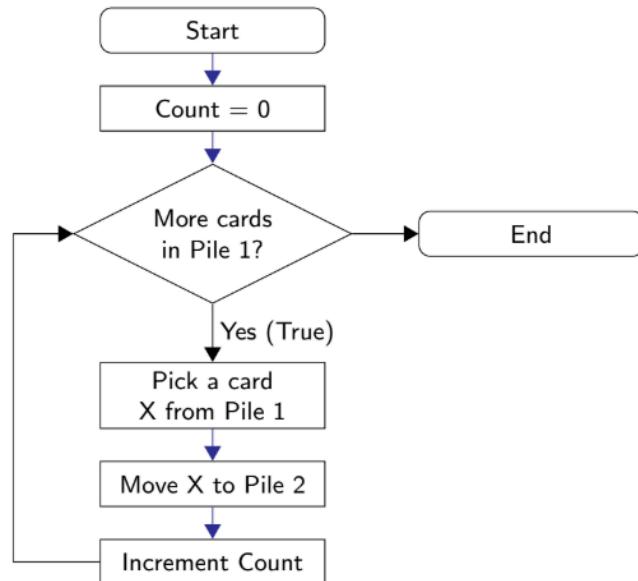


Explanation :

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
```

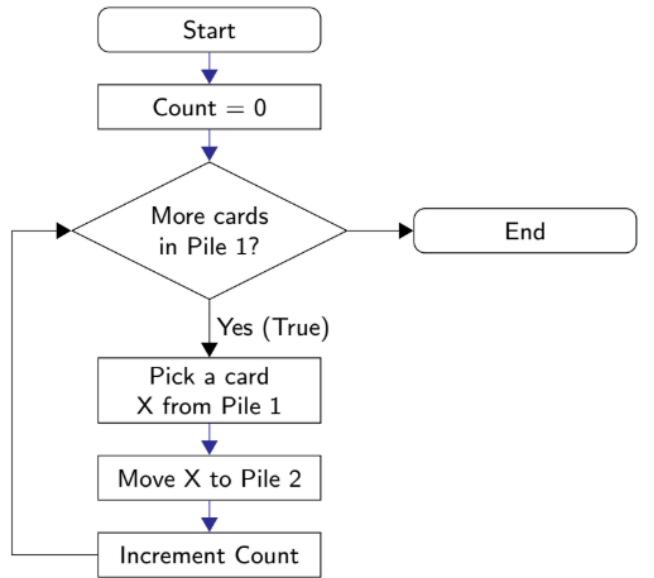
- 1 Assign a value to a variable



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
```

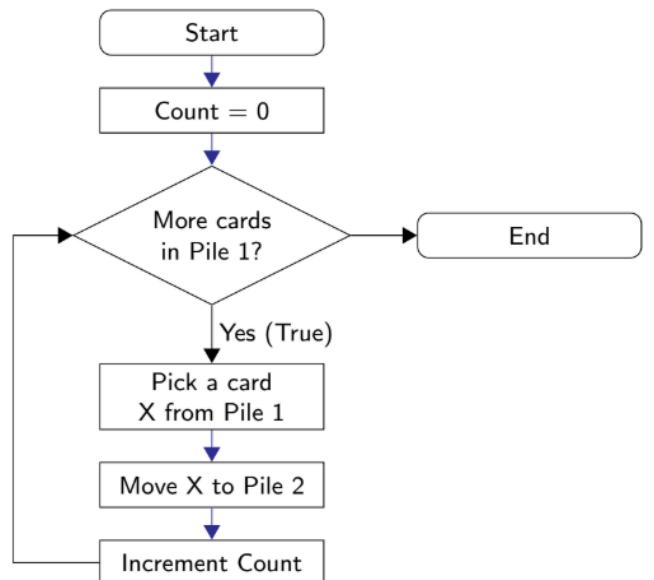
- 1 Assign a value to a variable
- 2 Repeat steps while condition holds



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

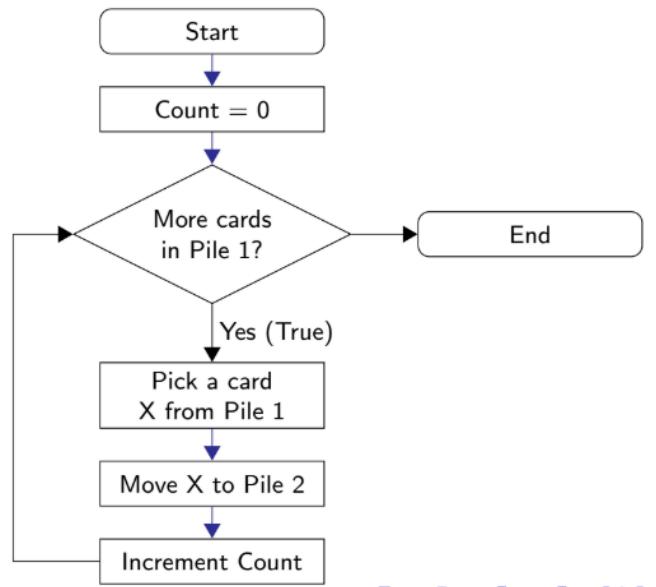
QN : 3

Lecture 10 : Pseudocode for iteration with filtering

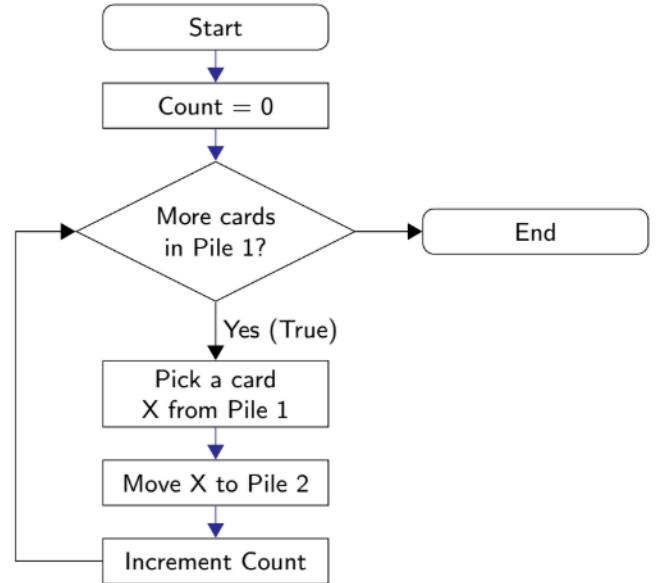
Pseudocode: Iteration and Filtering

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
```

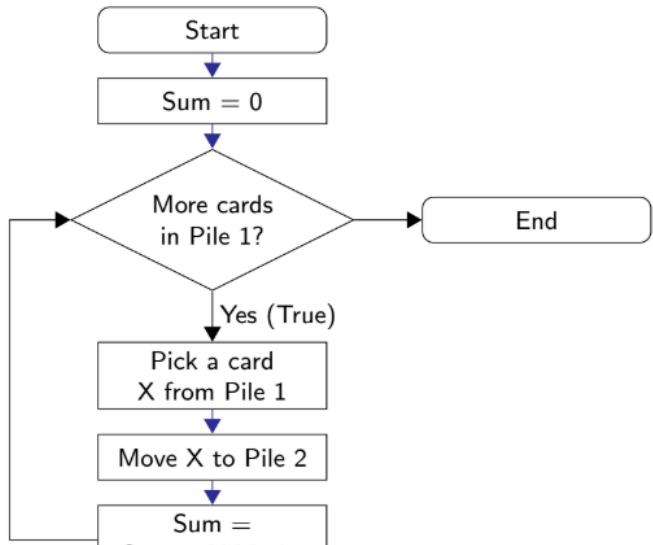


- Will dispense with Start and End, henceforth

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**



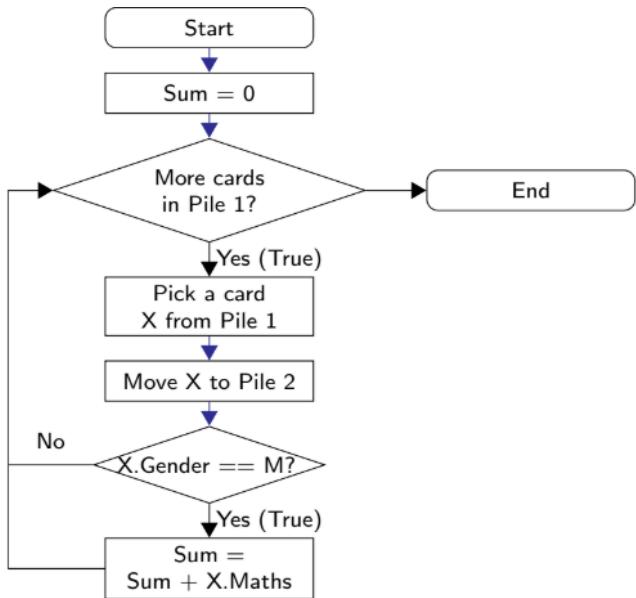
Sum of Boys' Maths marks

```

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 (`=`)

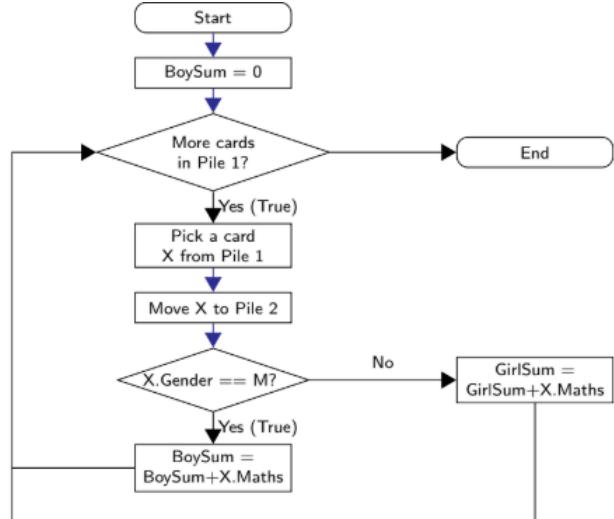


Sum of Boys' and Girls' Maths marks

```

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
    }
}

```



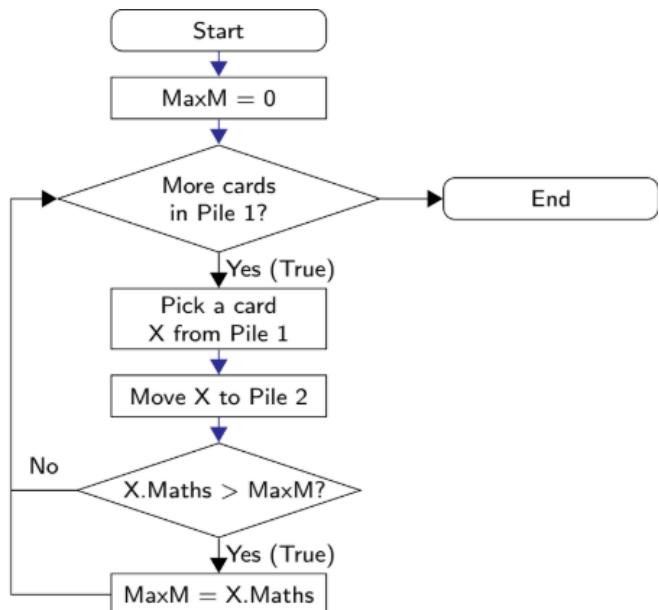
Alternative branch for conditional

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

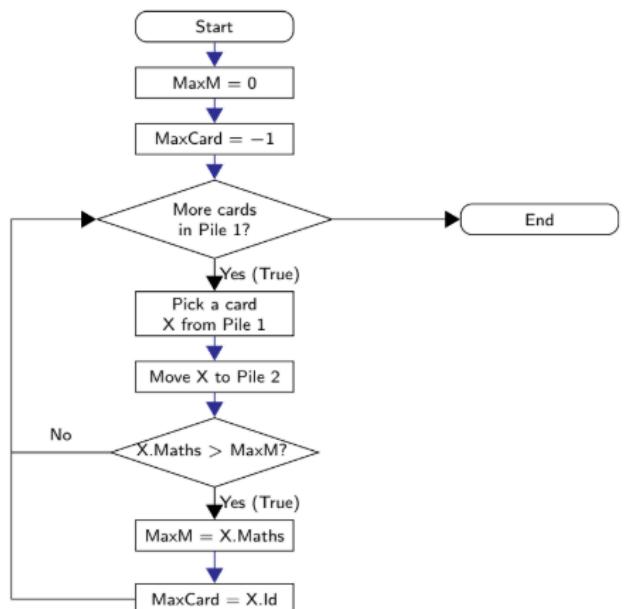
 if (**X**.Maths > **MaxM**) {

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  
}|
```

I

1st → Chennai Male

2nd → not like 1st

PA : 3,8,9,11

GA : 6,7,8

Atleast → min → x >= value

Atmost → max → x <= value