CSE 121 Lesson 9:
Conditionals

Matt Wang

Spring 2024

sli.do #cse121-9

TAs:	Andy	Anju	Archit	Arkita	Autumn	Christian
	Hannah H	Hannah S	Heather	Hibbah	Janvi	Jessie
	Jonus	Julia	Luke	Maria	Mia	Ritesh
	Shayna	Simon	Trey	Vidhi	Vivian	Gumball?

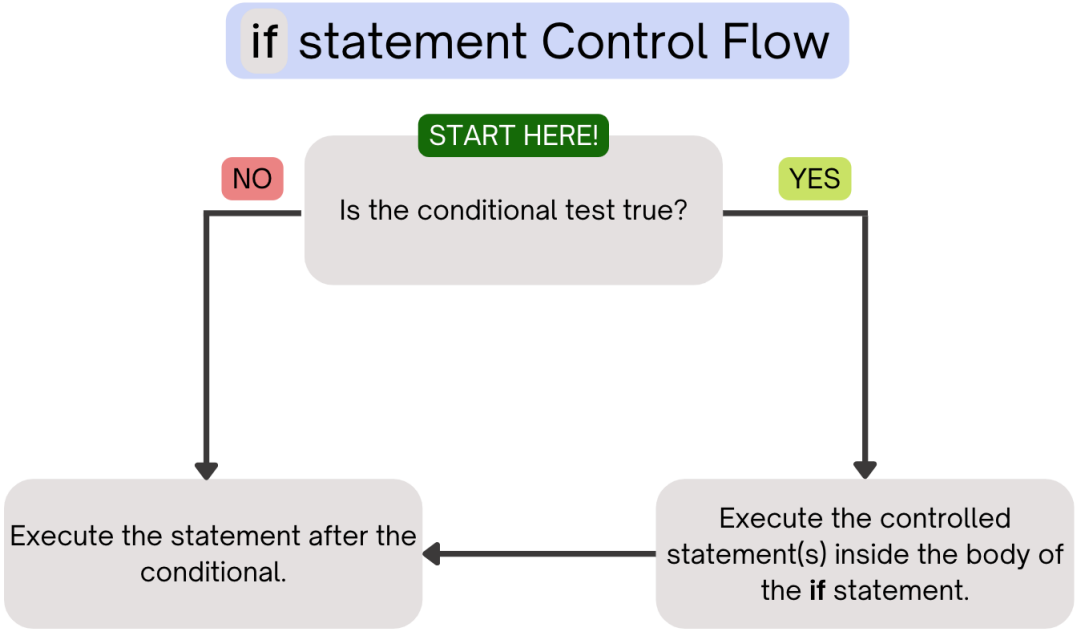
Today's playlist:
CSE 121 lecture beats 24sp

Announcements, Reminders

- Creative Project 2 released, due Thursday May 2nd
  - Note: uses Javadoc!
- R2 out yesterday, due Thursday May 2nd
  - Note: this is the last time C0 is eligible for resubmission!
- Mid-Quarter Formative Feedback with Ken Yasuhara for part of class on Wednesday, May 1st
- IPL tips!

(PCM) Conditionals (1/4)

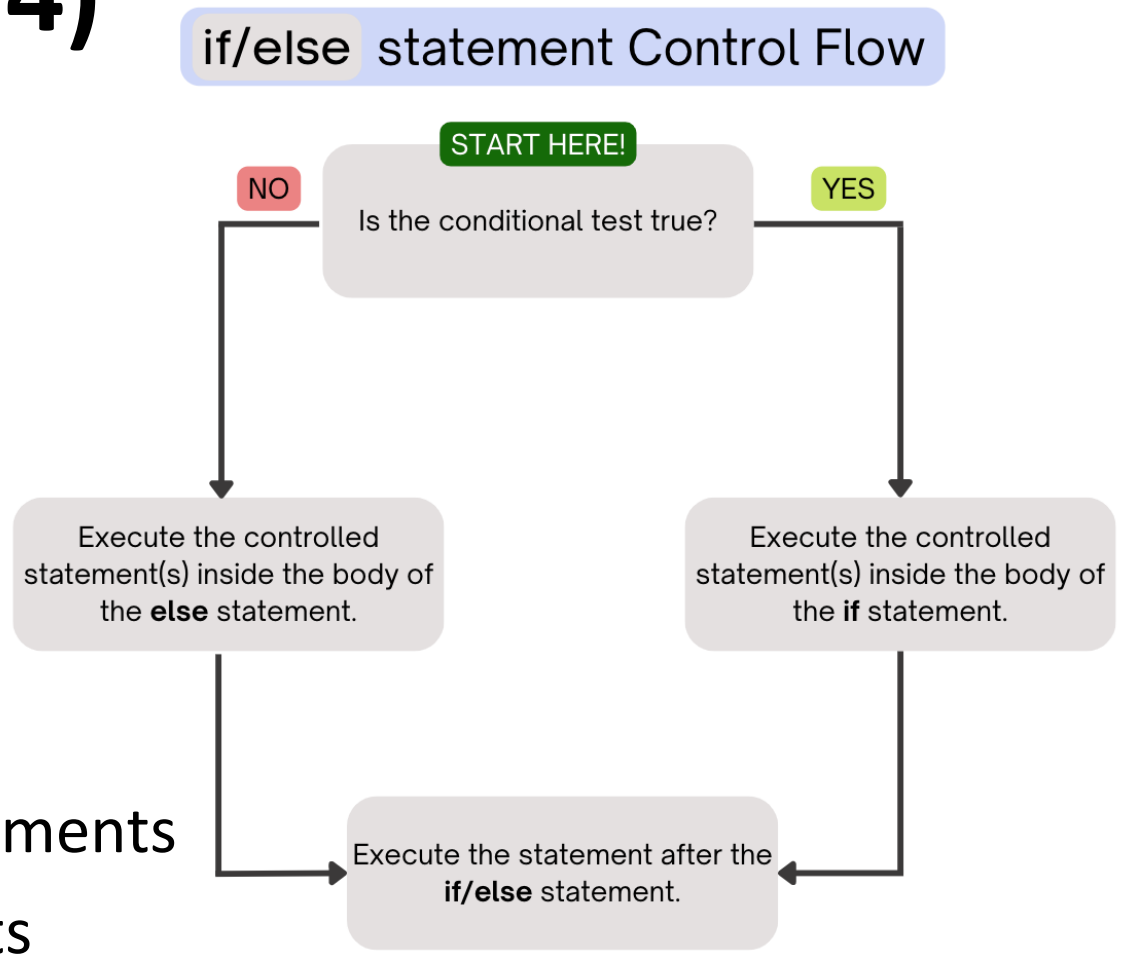
```
if (test) {
    body (statements to be executed)
}
```



Executes a block of statements
if and only if the test is true

(PCM) Conditionals (2/4)

```
if ( test ) {
        statement(s)
} else {
        statement(s)
}
```

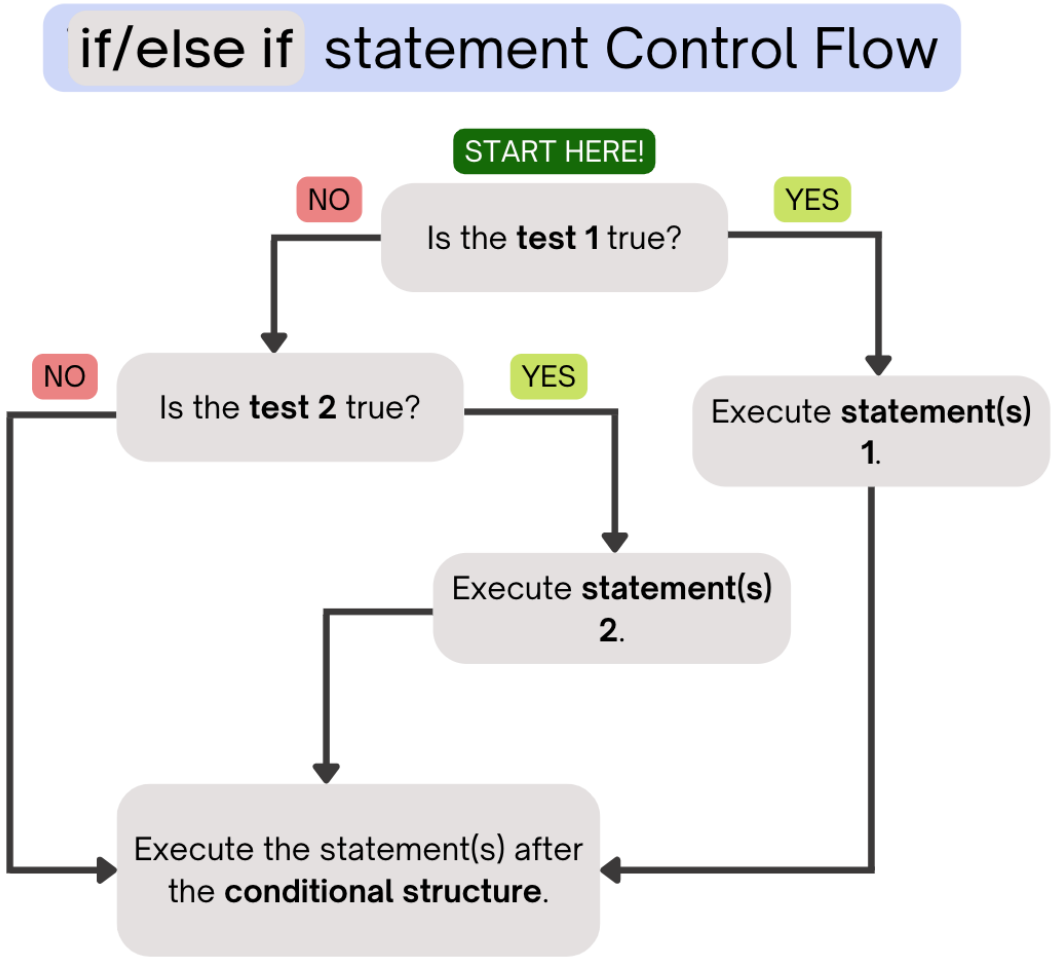


1. If the test is true: execute block of statements
2. If not, execute other block of statements

(PCM) Conditionals (3/4)

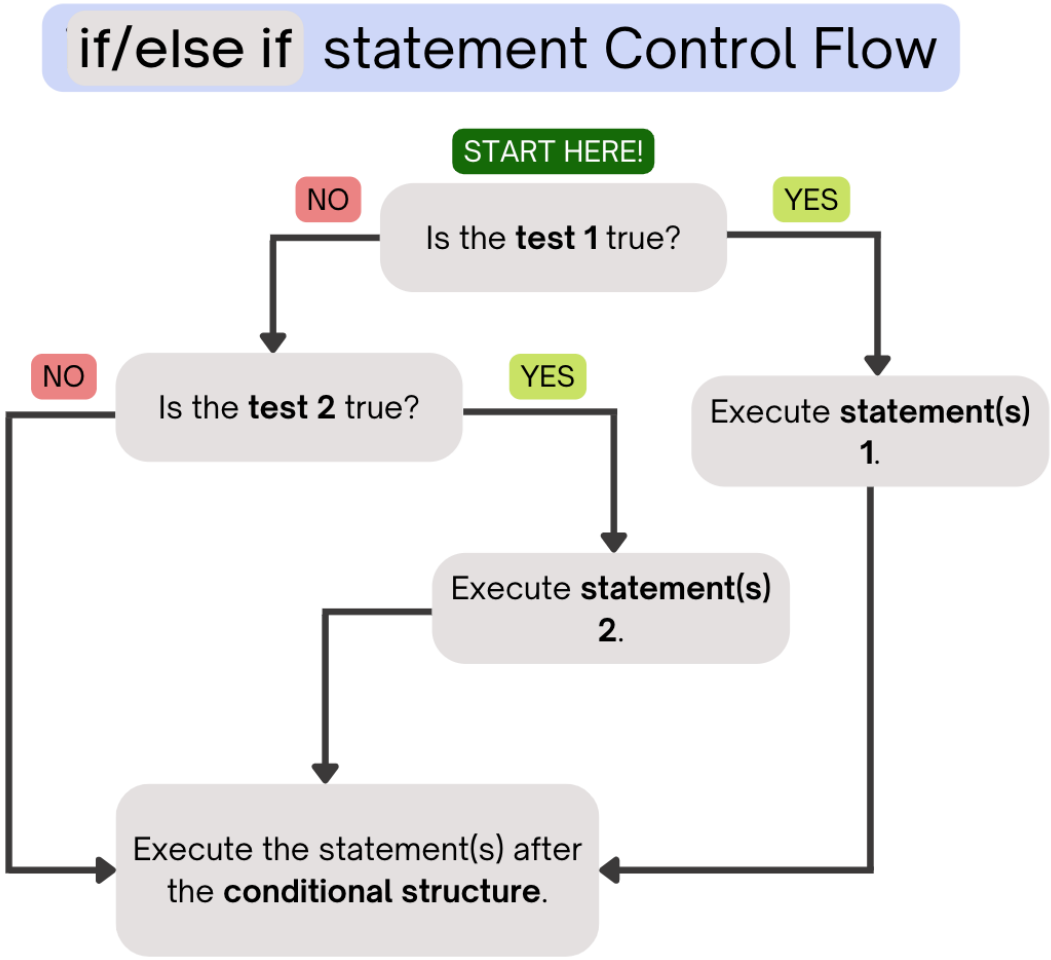
```
if ( test ) {
        statement(s)
} else if (test) {
        statement(s)
}
```

1. If the first test is true, execute that block
2. If not, proceed to the next test, and repeat
3. If none were true, don't execute any blocks



(PCM) Conditionals (4/4)

```
if ( test ) {
        statement(s)
} else if (test) {
        statement(s)
}
```



With a large if-else-if-else chain,

- if there is an ending else, exactly one block will execute
- if there is no ending else, zero or one blocks will execute

Poll in with your answer!

```
public static void main(String[] args) {
  for (int i = 1; i <= 3; i++) {
    System.out.print(mystery(i));
  }
}

public static String mystery(int n) {
  if (n % 2 == 1) {
    return "odd ";
  } else if (n == 1) {
    return "one ";
  }
  return "even ";
}
```

What does this program output?

A. odd even odd

B. one even odd

C. one even even

D. even even even

sli.do #cse121-9

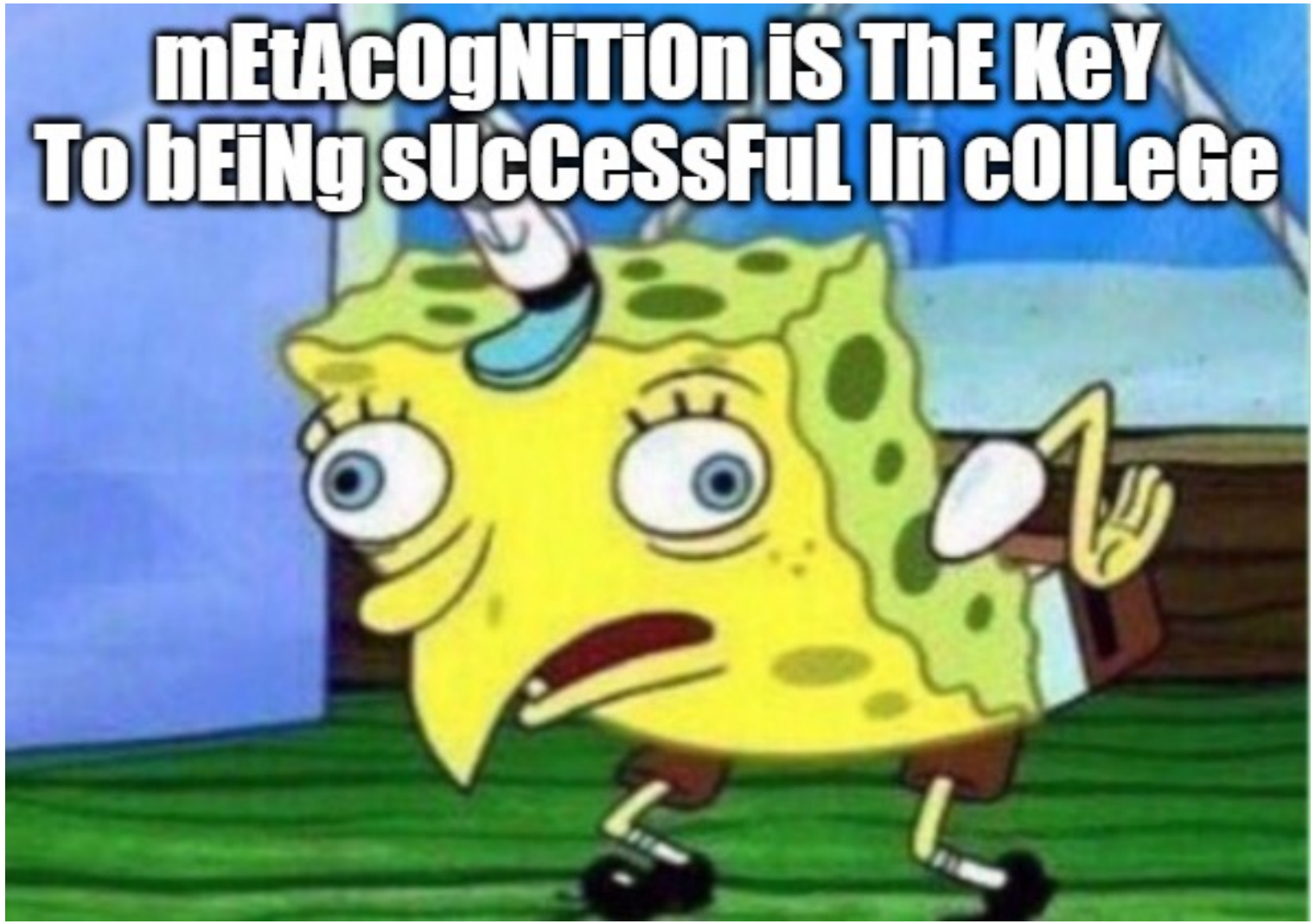
Poll in with your answer!

sli.do #cse121-9

```
public static void main(String[] args) {
  for (int i = 1; i <= 3; i++) {
    System.out.print(mystery(i));
  }
}

public static String mystery(int n) {
  if (n % 2 == 1) {
    return "odd ";
  } else if (n == 1) {
    return "one ";
  }
  return "even ";
}
```

This else if statement never runs!


mEtAcOgNiTiOn iS ThE KeY
To bEiNg sUcCeSsFuL In cOlLeGe

Common Problem-Solving Strategies

- **Analogy** – Is this similar to another problem you've seen?
- **Brainstorming** – Consider steps to solve problem before jumping into code
  - Try to do an example "by hand" → outline steps
- **Solve sub-problems** – Is there a smaller part of the problem to solve?
- **Debugging** – Does your solution behave correctly?
  - What is it doing?
  - What do you expect it to do?
  - What area of your code controls that part of the output?
- **Iterative Development** – Can we start by solving a different problem that is easier?

Common Problem-Solving Strategies

- **Analogy** – Is this similar to another problem you've seen?
- **Brainstorming** – Consider steps to solve problem before jumping into code
  - Try to do an example "by hand" → outline steps
- **Solve sub-problems** – Is there a smaller part of the problem to solve?
- **Debugging** – Does your solution behave correctly?
  - What is it doing?
  - What do you expect it to do?
  - What area of your code controls that part of the output?
- **Iterative Development** – Can we start by solving a different problem that is easier?

Common Problem-Solving Strategies

- **Analogy** – Is this similar to another problem you've seen?
- **Brainstorming** – Consider steps to solve problem before jumping into code
  - Try to do an example "by hand" → outline steps
- **Solve sub-problems** – Is there a smaller part of the problem to solve?
- **Debugging** – Does your solution behave correctly?
  - What is it doing?
  - What do you expect it to do?
  - What area of your code controls that part of the output?
- **Iterative Development** – Can we start by solving a different problem that is easier?

Food for Thought

This week's food for thought is:

- one of matt's favourite areas of computer science
- less related to tech & society than the others…
- also the most ambitious, so don't stress about it – sit back, enjoy the ride :)

Wouldn’t it be nice…

We’ve seen that some for loops go on forever:

```
for (int i = 0; i < 10; i--) {
  System.out.println(i);
}
```

```
for (;true;) {

}
```

Wouldn’t it be nice if Java (or “the compiler”) could catch this for us? I mean, the loop “obviously” never ends…

The Halting Problem (1/2)

Benedict Cumberbatch showed that it's impossible to generally solve this problem.

Regardless of:

- how good (big, fast) your computer is
- how good your algorithm is
- **what people come up with the future!**

Given a Java program, it is impossible to always know if it eventually stops (or loops infinitely).

The Halting Problem (2/2)

~~Benedict Cumberbatch~~ **Alan Turing** showed that it's impossible to generally solve this problem.

Regardless of:

- how good (big, fast) your computer is
- how good your algorithm is
- **what people come up with the future!**

Given a Java program, it is impossible to always know if it eventually stops (or loops infinitely).

Alan Turing at 24 (1936). He had a storied (if also very tragic and short) life.

Many, many problems are unsolvable.

I don't mean "we currently don't know how to solve them".

I mean, **"there is no algorithm that will ever solve them"**.

Here are some related "**undecidable**" problems:

- given a Java program, are all the types correct?
- given a polynomial equation, does it have integer solution(s)?
- given any Magic: The Gathering board,
  does either player have a guaranteed winning strategy?

“This statement is false”

In fact, there’s an even more concerning result: there is at least one math statement that we can’t prove true or false.

What is that statement? It looks something like…

“This statement is false”.

In search of perfection

Even though we know it's "impossible", we still:

- try avoiding infinite loops
- type-check our Java programs
- play Magic: The Gathering (?)
- try to prove things in (and do) math!

Dessert for Thought!

I argue there are two takeaways:

1. Don't let perfection be the enemy of the good!
   - applies to you in CSE 121 and as a programmer :)
   - fundamental basis of much of computer science
2. Like thinking about these sorts of problems?

   <u>This is also computer science!</u>

   (not all CS is just coding…) See: CSE 311, CSE 417/431