Introduction to Computers for Engineers:

Recitation #3

Learning Objectives

- Understand the use of logical and comparative statements
- Understand the structure of if/else/elseif statements
- Understand how to use if/else/elseif structures and logical/comparative statements to have your program make decisions via branching
- Understand how you can use a visualization to help you figure out how a program should branch

Logical and Comparative Operators

Operator	Syntax	True When
AND	`A&&B`	A and B are both true
OR	`A||B`	A, B, or both A and B are true
NOT	`~A`	A is false
Equal To	`X == Y`	X and Y have the same value
NOT Equal To	`X ~= Y`	X and Y have different values
Greater Than	`X > Y`	X has a higher value than Y
Less Than	`X < Y`	X has a lower value than Y
Greater Than OR Equal To	`X >= Y`	X has a higher or equal value to Y
Less Than OR Equal To	`X <= Y`	X has a lower or equal value to Y

Activity 1: Warm-up

- Let's get used to conditional statements!
- Write a function called **recitation3** that takes 2 inputs, x and y:
- Has 1 output, z
  - When x is 0, z should be 0
  - Otherwise, when y is 0, z should be 1
  - Otherwise, z should be 2

- **Discuss: How many test cases should you write?**
- **Discuss: What values might be most useful in your test cases?**

Activity 1: Solution

```
function [z] = recitation3(x, y)

if x == 0
    z = 0;
elseif y == 0
    z = 1;
else
    z = 2;

end
```

Activity 2: Divide and Solve

Team 1 Code:

- Write a function **team1** that takes 2 inputs, x and y
- Has 1 output, z
  - When both x and y are positive, and x is greater than y, z should be 4
  - When both x and y are positive and x is less than or equal to y, z should be 3
  - When both x and y are negative, and x is greater than y, z should be 2
  - When both x and y are negative, and x is less than or equal to y, z should be 1
  - Otherwise, z should be 0

Team 2 Code:

- Write a function **team2** that takes 2 inputs, x and y
- Has 1 output, z
  - When x and y have different signs, z should be 0
  - When both x and y are negative, and y is greater than or equal to x, z should be 1
  - When both x and y are negative, and y is less than x, z should be 2
  - When both x and y are positive, and y is greater than or equal to x, z should be 3
  - Otherwise, z should be 4

As a whole team, come up with test cases. Use the same test cases for both programs. What are the results of these tests?

Activity 2: Solution

Team 1 Code:

- Write a function **team1** that takes 2 inputs, x and y
- Has 1 output, z
  - When both x and y are positive, and x is greater than y, z should be 4
  - When both x and y are positive and x is less than or equal to y, z should be 3
  - When both x and y are negative, and x is greater than y, z should be 2
  - When both x and y are negative, and x is less than or equal to y, z should be 1
  - Otherwise, z should be 0

```
function [z] = team1(x, y)

if (x>0 && y>0) && (x > y)
    z = 4;
elseif (x>0 && y>0) && (x <= y)
    z = 3;
elseif (x<0 && y<0) && (x > y)
    z = 2;
elseif (x<0 && y<0) && (x <= y)
    z = 1;
else
    z = 0;

end
```

As a whole team, come up with test cases. Use the same test cases for both programs. What are the results of these tests?

Activity 2: Solution

```
function [z] = team2(x, y)

if (x>0 && y<0)||(x<0 && y>0)
    z = 0;
elseif (x<0 && y<0) && (y>=x)
    z = 1;
elseif (x<0 && y<0) && (y<x)
    z = 2;
elseif (x>0 && y>0) && (y>=x)
    z = 3;
else
    z = 4;
end
```

Team 2 Code:

- Write a function **team2** that takes 2 inputs, x and y
- Has 1 output, z
  - When x and y have different signs, z should be 0
  - When both x and y are negative, and y is greater than or equal to x, z should be 1
  - When both x and y are negative, and y is less than x, z should be 2
  - When both x and y are positive, and y is greater than or equal to x, z should be 3
  - Otherwise, z should be 4

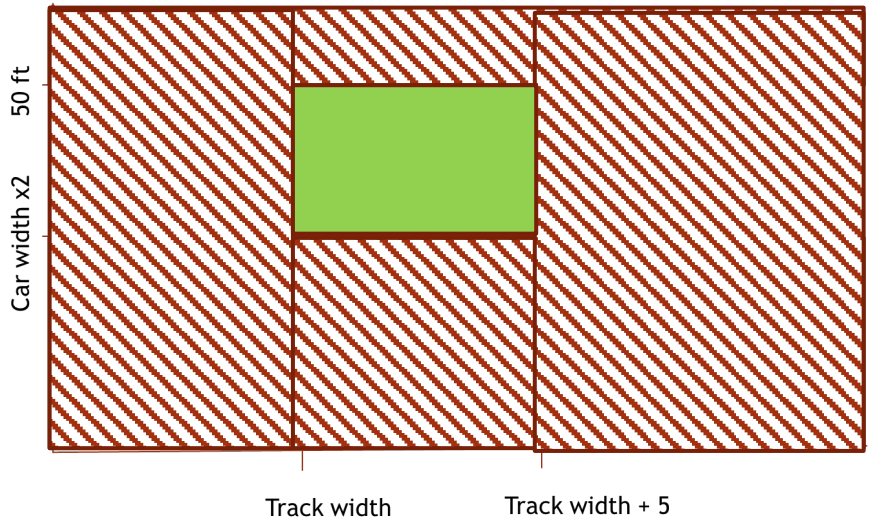
As a whole team, come up with test cases. Use the same test cases for both programs. What are the results of these tests?

Example: Train Cars

- We are writing a program as a pre-screener for approving designs for train cars. Different systems use different track widths, but we want a program that will enforce the following general rules for us:
  - Train cars must be as wide as the train tracks
  - Train cars cannot be more than 5 feet wider than the train tracks
  - Train cars must be more than twice as long as they are wide
  - Train cars cannot exceed 50 feet

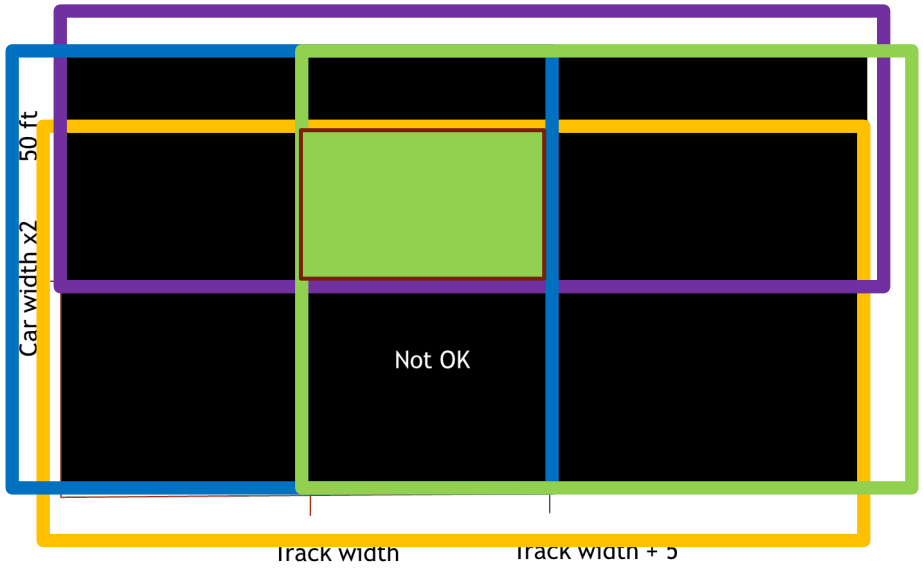
Example – Train Cars – Decision Space

Example – Train Cars – Method 1 – Rejection Criteria




```
function ok = trainCars(trackW,carW,carL)


if carL > 50
    ok = false;
elseif carL < carW*2
    ok = false;
elseif carW < trackW
    ok = false;
elseif carW > trackW + 5
    ok = false;
else
    ok = true
end
```

Example – Train Cars – Method 2 – Acceptance Criteria



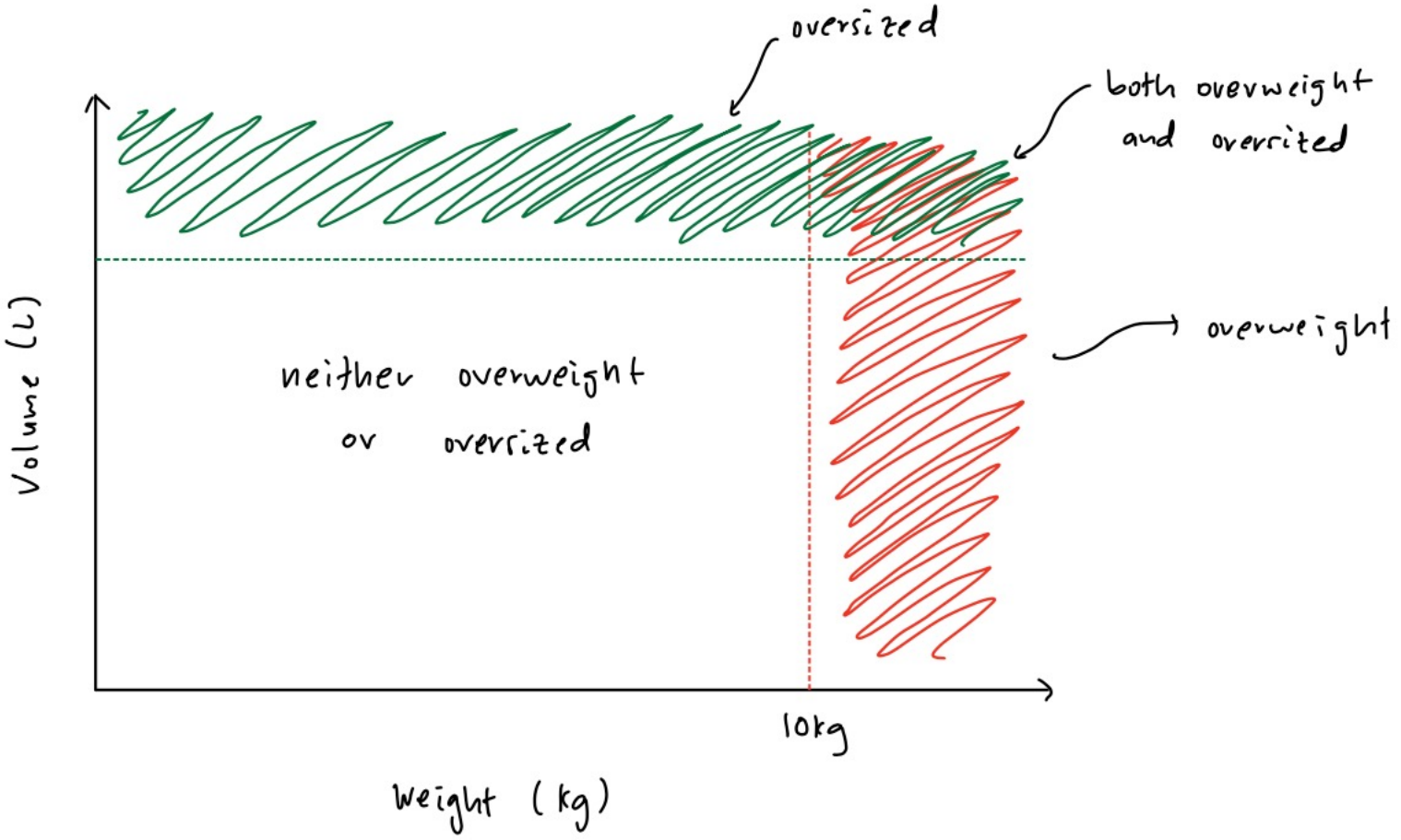

```
function ok = trainCars(trackW,carW,carL)


if carL <= 50 && carL >= carW*2...
    &&carW >= trackW && carW <= trackW+5
    ok = true;
else
    ok = false
end
```

Activity 3: Visualizing if-else statements

- Let's say we are calculating postage rates for packages
- We want to charge for both weight and volume
  - $3.00 per kilogram
  - $1.00 per liter (0.001 cubic meter)
- We also want surcharges
  - $10.00 if the item weighs more than 5 kg
  - $8.00 if the item has a volume of more than 20 liters (0.02 cubic meters)

- Think about the boundaries of this space based on different price equations
- Use the whiteboard
- Try to draw a 2D chart of regions where:
  - The package is neither overweight or oversized
  - The package is oversized
  - The package is overweight
  - The package is both overweight and oversized
- **Discuss: When should you use > or < vs >= or <= to differentiate between regions?**

Activity 3: 2D Visualization



Activity 4: Implementation

- Translate your charts into code!
- Create a new function called **computePostage** that calculates postage given inputs of weight and volume
- How does the image you created translate into code?
  - Which comparative operators did you use?
  - How many different branches (if/elseif/else keywords) do you need?
- **Come up with test cases within your group.**
- **Discuss: What particular values did you want to test and why?**

Activity 4: Solution

```
function [cost] = computePostage(weight, volume)

% check if both oversized and overweight
if (weight > 5) && (volume > 20)
    cost = (weight*3) + volume + 10 + 8;

% check if oversized
elseif (weight > 5)
    cost = (weight*3) + volume + 10;

% check if overweight
elseif (volume > 20)
    cost = (weight*3) + volume + 8;

else
    cost = (weight*3) + volume;
end
```