CS1010 Tutorial 2
Group BC1A

3 September 2020

Topics for today

Objectives

- Recap on Topics (Functions, First C Program, Conditional Statement)
- Going through problem set 3, 5, 8
- Assignment 2
- Summary

Recap on Functions (Unit 3)

Allows for reusability of code.

What is a function?

- Function is something that takes in an input, process it and returns an output
- Function allows for breaking down of complex task into smaller sub-task for computation

Recursive functions

- This is a function that calls itself, creating a recursive tree until termination
- To create a recursive function, always find the **BASE CASE** first
  - The base case is a case that will always run within each call of the function
- After finding the base case, find the **TERMINATING CONDITION**
  - The termination condition is a condition that stops the recursive tree

Problem 3.1 Question

The mean absolute deviation, or MAD, of a set of integers measures how spread out a set of data is. The *absolute deviation* is the absolute difference between an element in the list with the mean of values of the list. The mean absolute deviation is the mean of all the absolute difference. In other words, given $L = \{l_0, \ldots l_{k-1}\}$, the MAD of L is:

$$\frac{\sum_{i=0}^{k-1} |l_i - \mu|}{k}$$

$l_i - \mu$

How do you find MAD by composing various functions we have seen? Do you need a new function?

- μ = mean of each value. → $\frac{\text{sum of all } L}{\text{no of items.}}$
- $|l_i - \mu|$ = diff between each term and the mean
- Σ = sum of all the differences.

Problem 3.1 Solution

Main goal of this question is to break down a problem into smaller parts

We can break this question down into 4 steps

- first find the mean with the function mean(L, k)
- then subtract mean away from every element in the list with subtract.
- then find the abs of every element (this is a new method that is needed)
- then find the mean of the elements again.

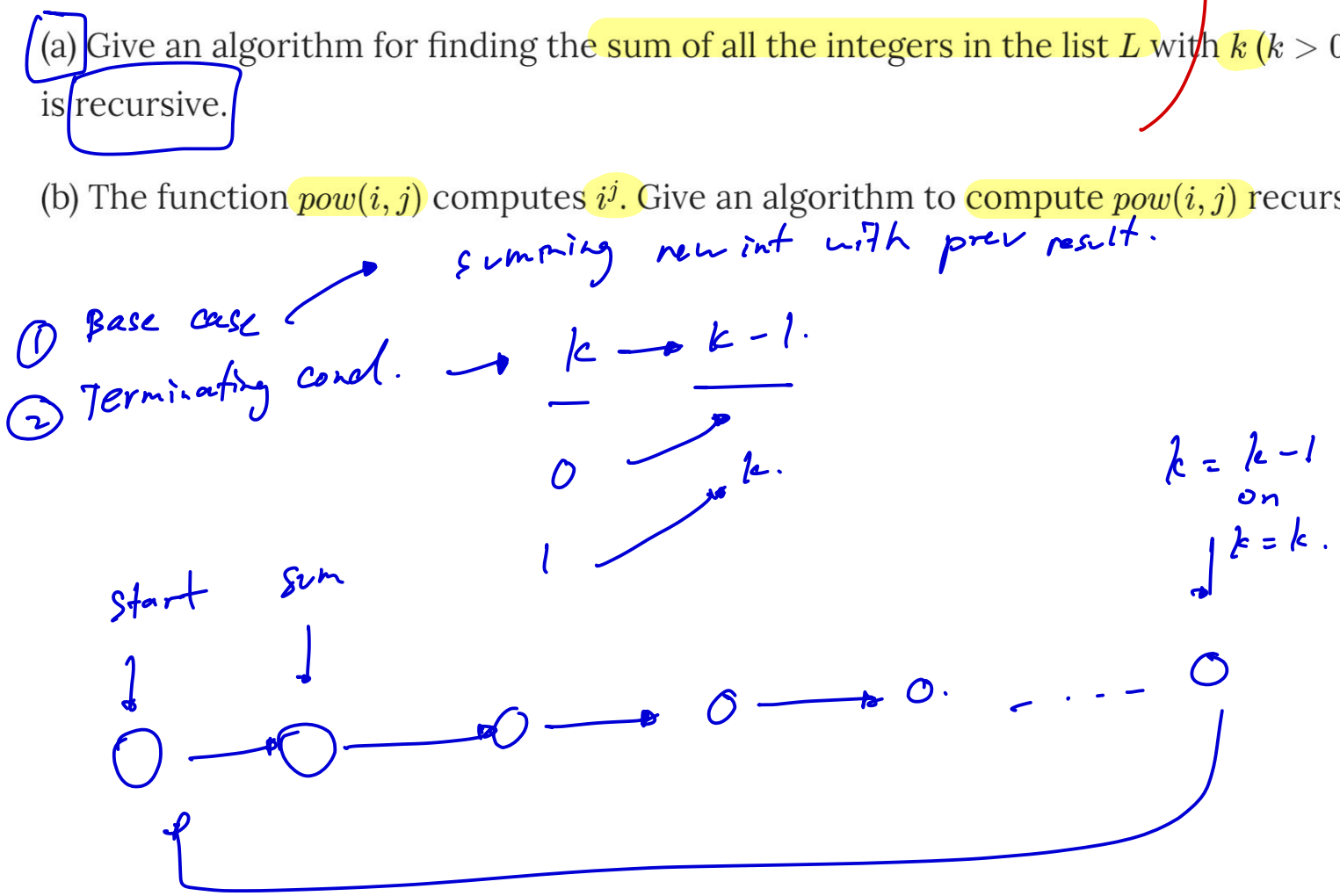
We can compute MAD by finding the mean(abs(subtract(L, k, mean(L, k))), k)

We need a new function that takes in a list of numbers and return a list containing the absolute values of the numbers.

Problem 3.2 Question

(a) Give an algorithm for finding the sum of all the integers in the list L with k ($k > 0$) integers that is recursive.

(b) The function $pow(i, j)$ computes i^j. Give an algorithm to compute $pow(i, j)$ recursively.



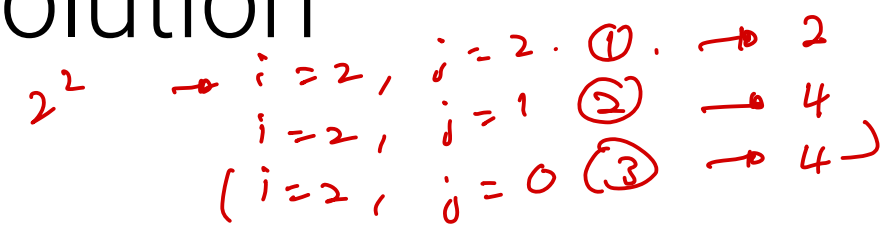
Problem 3.2(a) Solution

- Base case
  - Add the current value into the total sum until this point
- Terminating condition
  - When there are no more values to add, which is when current index > number of data
- Implementation
  - Create a function sum(L, i, j) that computes the sum of elements Li to Lj
    - i is the Starting index
    - j is the Ending index
    - L is the total sum at index i
  -

long values = [ ];
for (i=0 ; i < values.size; i+=1)
sum += values[i];

```
long sum(long L, long i, long j) {
        long values[10] = {1,2,3,4,5,6,7,8,9,10};

        if (i < j) {
                long currSum = L + values[i];
                return sum(currSum, i + 1, j);
        } else {
                return L;
        }
}
```

Problem 3.2(b) Solution



- Base case
  - Multiply i x i
- Terminating condition
  - When there are no need to multiply anymore,, which is when current index > j
- implementation
  - Create a function power(i, j) that computes power i^j
    - i is the Number to find the power of
    - j is the power

```
long power(long i, long j) {
        if (j > 0) {
                return i * power(i, j - 1);
        } else if (j < 0){
                return (1/i) * power(i, j + 1);
        } else if (j == 0) {
                return 1;
        }
}
```

Problem 5.1(a) Question

```
1  #include <math.h>
2  long square(long x)
3  {
4    return x * x;
5  }
6
7  double hypotenuse_of(long base, long height)
8  {
9    return sqrt(square(base) + square(height));
10 }
11
12 int main()
13 {
14   hypotenuse_of(3.0, 4.0); // <- use 3.0 and 4.0 instead of int.
15 }
```

Java, c++, C ?

floating type →

4.0000032 + 0.0000014

= 4.0000172

≈ 4.000018

implicit typecasting: (anguage auto typecast
without telling the user,

typecast
↓
changing the
type

(float → int (long))

hypo_of (3.2 , 4.1).

3 4

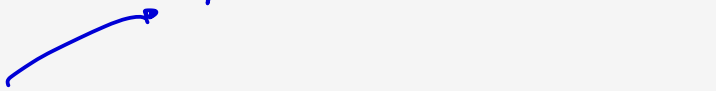
→ hypo_of (3, 4).

int (long) → double.

3 → 3.000000

Problem 5.1(b) Question

```c
#include <math.h>
long square(long x)
{
  return x * x;
}

double hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}

int main()
{
  long h = hypotenuse_of(3, 4); // <-- assign to a long variable
}
```



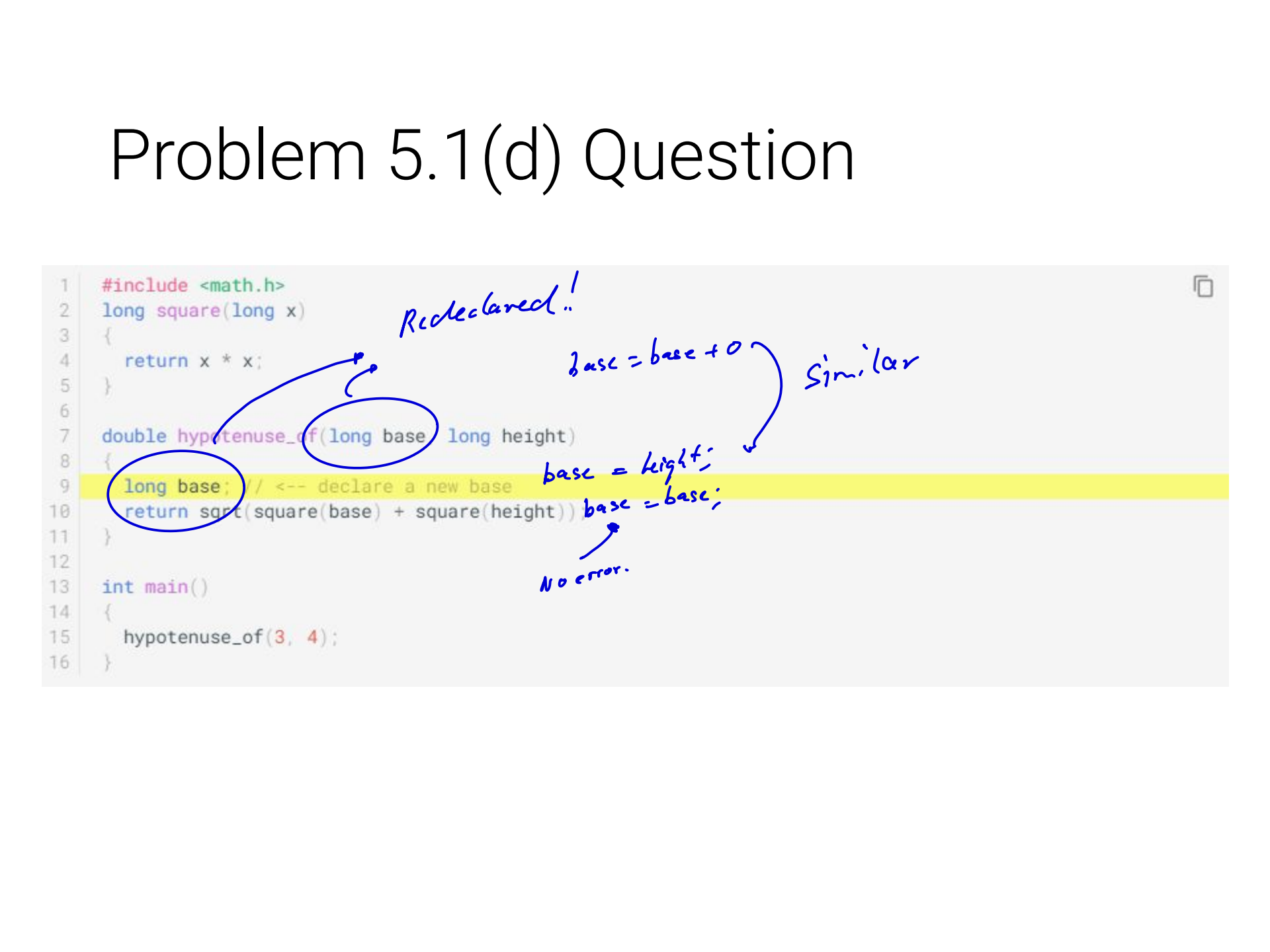
Problem 5.1(c) Question

```c
#include <math.h>
long square(long x)
{
  return x * x;
}

double hypotenuse_of(long base, long height)
{
  return sqrt(square(base) + square(height));
}

int main()
{
  Hypotenuse_Of(3, 4); // <-- use a different case
}
```

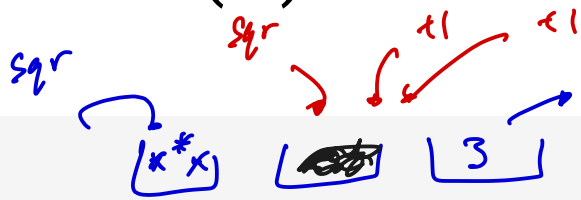
in C, everything is case sensitive!!!

Problem 5.1(d) Question

```
1  #include <math.h>
2  long square(long x)
3  {
4    return x * x;
5  }
6
7  double hypotenuse_of(long base, long height)
8  {
9    long base; // <-- declare a new base
10   return sqrt(square(base) + square(height));
11 }
12
13 int main()
14 {
15   hypotenuse_of(3, 4);
16 }
```



Problem 5.1(e) Question




```
1  #include <math.h>
2  long square(long x)
3  {
4    long sqr = x * x; /// <-- declare and assign in one go
5    return sqr;
6  }
7
8  double hypotenuse_of(long base, long height)
9  {
10   return sqrt(square(base) + square(height));
11 }
12
13 int main()
14 {
15   hypotenuse_of(3, 4);
16 }
```

long sqr;
sqr += 1
output sqr = [illegible]

long sqr;
sqr = 3;
output sqr = 3.

Problem 5.1(f) Question

```
1  #include <math.h>
2
3  long sqr;  // <-- use global variable
4  long square(long x)
5  {
6    sqr = x * x;
7    return sqr;
8  }
9
10 double hypotenuse_of(long base, long height)
11 {
12   return sqrt(square(base) + square(height));
13 }
14
15 int main()
16 {
17   hypotenuse_of(3, 4);
18 }
```

Problem 5.1(g) Question

```c
#include <math.h>

int main()
{
  long square(long x) // <-- define function within function.
  {
    return x * x;
  }

  double hypotenuse_of(long base, long height) // <--
  {
    return sqrt(square(base) + square(height));
  }

  hypotenuse_of(3, 4);
}
```

illegal in C.

Problem 8.1 Question

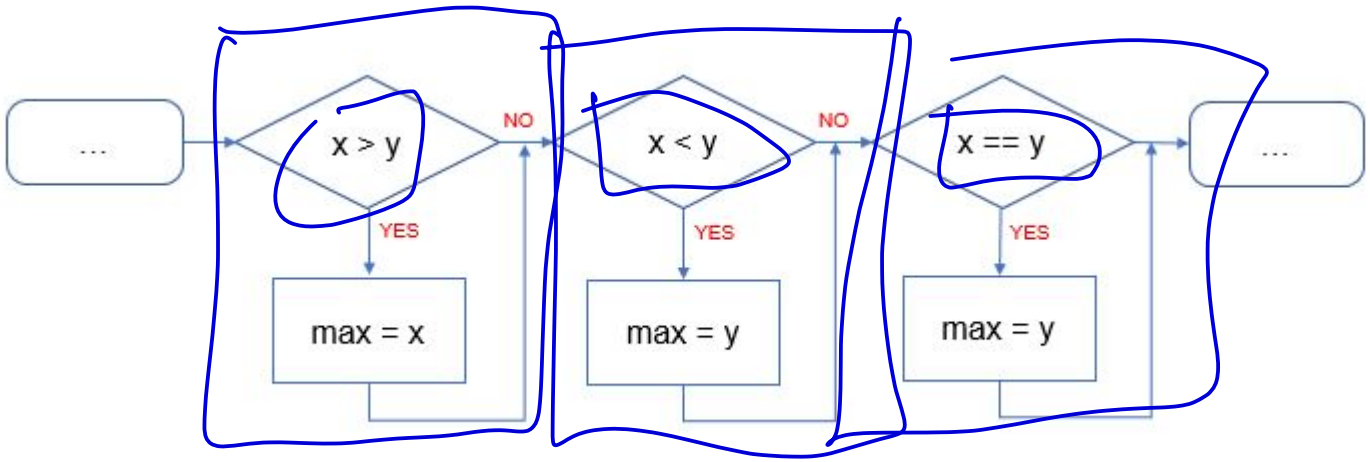
```
1  long factorial(long n)
2  {
3    long answer;
4    if (n == 0) {
5      return 1;
6    }
7    answer = n * factorial(n - 1);
8    return answer;
9  }
```

and

```
1  long factorial(long n)
2  {
3    long answer;
4    if (n == 0) {
5      answer = 1;
6    } else {
7      answer = n * factorial(n - 1);
8    }
9    return answer;
10 }
```

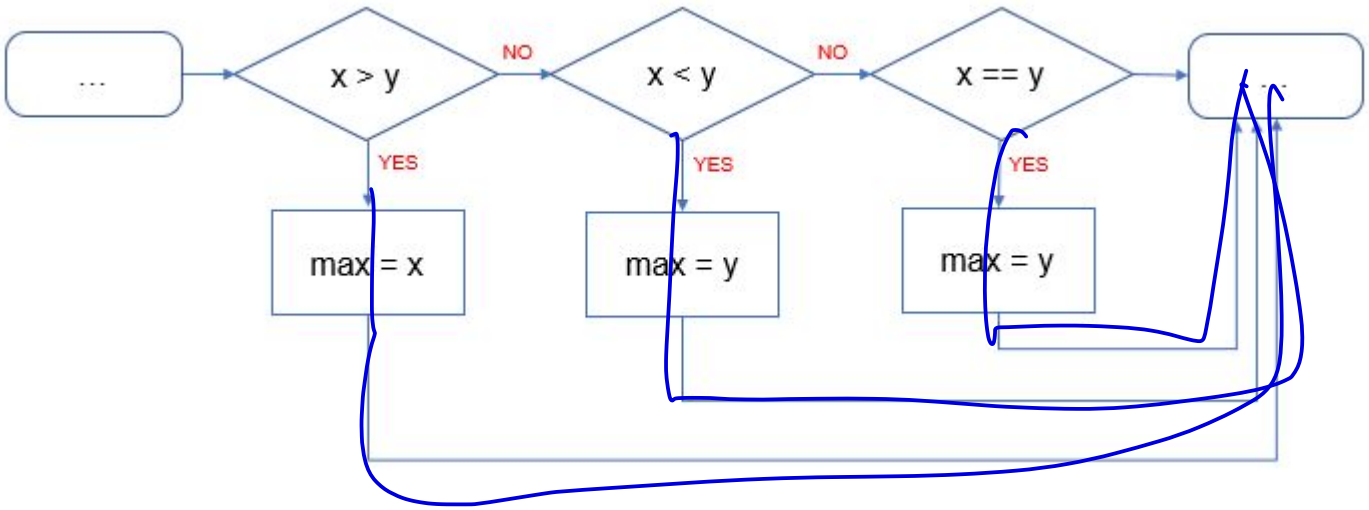
Problem 8.2(a) Question & Answer

```
1  if (x > y) {
2    max = x;
3  }
4  else if (x < y) {
5    max = y;
6  }
7  else if (x == y) {
8    max = y;
9  }
```



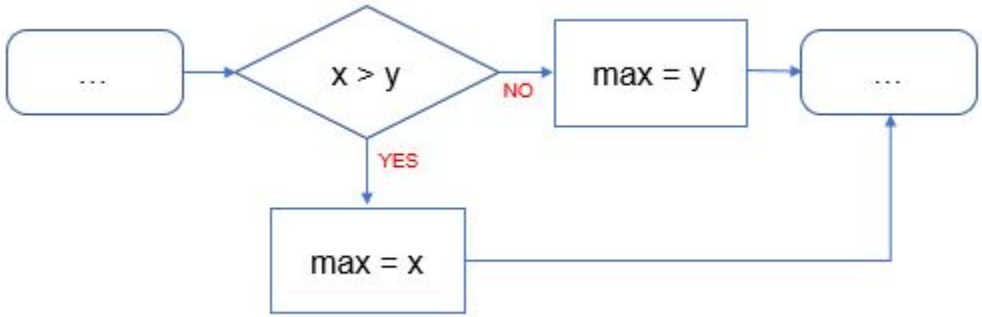
Problem 8.2(b) Question & Answer

```
1  if (x > y) {
2    max = x;
3  } else if (x < y) {
4    max = y;
5  } else if (x == y) {
6    max = y;
7  }
```



Problem 8.2(ㄷ) Question & Answer

```
if (x > y) {
  max = x;
} else {
  max = y;
}
```



Problem 8.3 Answer

```
if (score >= 5) {
        // table 3
        if (score >= 8) {
                // A
        } else {
                // B
        }
} else {
        // table 4
        if (score >= 3) {
                // C
        } else {
                // D
        }
}
```

Assignment 1

- https://nus-cs1010.github.io/2021-s1/as01.html
- Tasks
  - Box
  - Digits
  - Suffix
  - Taxi

Assignment 1

- Ceil function for Long

```
// Every x m (or less) costs c
long lceil(long distance, long x) {
  long result = distance / x;
  if (distance % x != 0) {
    result += 1;
  }
  return result;
}
```