Marcus Moquin – COMP1405 – September 9th, 2020

Problem 1

Decomposition

- Since decomposition is breaking down the problem into smaller parts, those parts being smaller, simpler problems to solve, we can use this in taking a shower.
- In this example, going to the washroom, turning the nozzle to run the water, adjust the nozzle to the temperature of water you wish, undress yourself, get into the shower, clean yourself, turn the nozzle back to off, get out of the shower, get a towel and dry yourself off.

Pattern Recognition

- Pattern recognition makes you notice similarities between different processes and data, which could lead to reusing solutions because their similarities show that these different processes could be done the same way as other processes.
- An example of this is fingerprint scanning to unlock certain things like your smartphone. After one quick setup, registering your fingerprint in your phone's data, every single time you try to unlock your phone with your fingerprint, your phone tries to process the fingerprint and if they find your exact fingerprint in the data, it'll unlock your phone since it recognizes it.

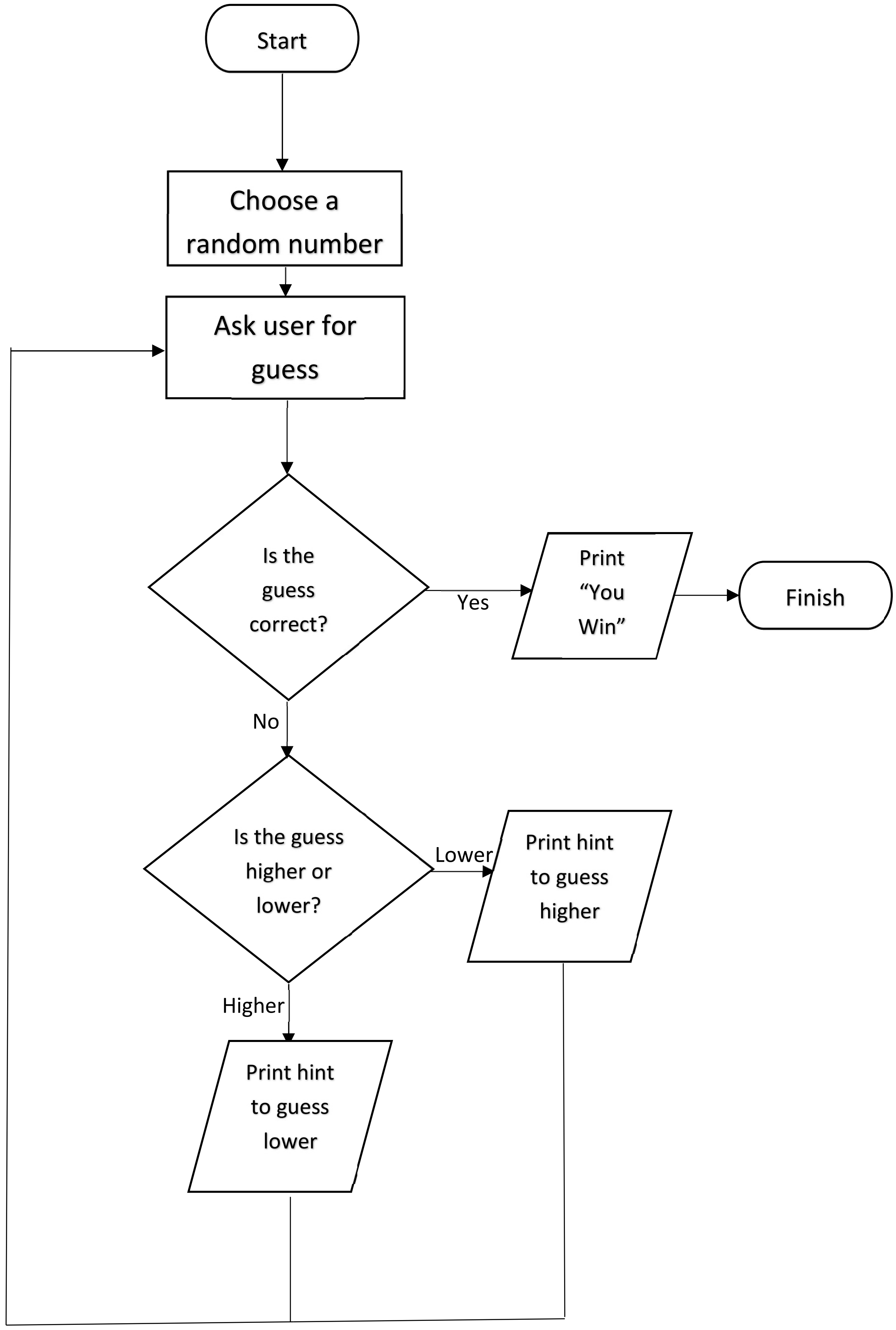
Abstraction

- Abstraction takes things out that are not as important as others, leading to simpler plans.
- Example here is baking a cake, we know it has general characteristics such as, needing ingredients, needing a specific amount of those ingredients and timings for baking and such. Since abstraction takes out the specifics, it takes out the specifics like what ingredients we need, how much of those ingredients and the timing and temperature for baking the cake itself.

Algorithm Design

- Algorithm design means we use our tools to produce an unquestionable, ordered set of instructions which should be able to be applied generally.
- With that being said, we use pseudocode to write out in plain English what it needs to do, what it should do and how it should work, and then we write that pseudocode into programming language so that the computer can eventually understand it and use the algorithm to perform the function properly.
- We can have an issue, write pseudocode on how to fix it and instead of rewriting in programming language, we can read the pseudocode and fix the issue with it since we wrote out steps on how to fix it.

Problem 2


Start
Choose a random number
Ask user for guess
Is the guess correct?
Yes
Print "You Win"
Finish
No
Is the guess higher or lower?
Lower
Print hint to guess higher
Higher
Print hint to guess lower

Problem 3

```
1. The program chooses a random number.
2. The program asks the user for a guess.
3. For each guess given:
    3.1. If the guess is wrong:
        3.1.1. If the guess is higher than the generated number:
            3.1.1.1. Print a hint to tell the user to guess lower.
        3.1.2. If the guess is lower than the generated number:
            3.1.2.1. Print a hint to tell the user to guess higher.
4. Output "You Win"
```

Problem 4

First guess: 10

Testing

Program Generated # = 42

Guess = 10

10 = 42? No.

10 < 42 or 10 > 42? 10 < 42.

“Guess Higher.”

Second guess: 20

Testing

Program Generated # = 42

Guess = 20

20 = 42? No.

20 < 42 or 20 > 42? 20 < 42.

“Guess Higher.”

Third guess: 50

Testing

Program Generated # = 42

Guess = 50

50 = 42? No.

50 < 42 or 50 > 42? 50 > 42.

“Guess Lower.”

Fourth guess: 25

Testing

Program Generated # = 42

Guess = 25

25 = 42? No.

25 < 42 or 25 > 42? 25 < 42.

“Guess Higher.”

Fifth guess: 42

Testing

Program Generated # = 42

Guess = 42

42 = 42? Yes.

“You Win!”