PLTW 1.3.4 Nested Branching & Input

PART 1:

ANSWER:

2a. 17

2b. 15: Orange
 17: Apple, Banana
 20: Potato
 22: Anything else inputted

2c. Because Bananas is also in Fruits so it never makes it to the else when inputted

VOCAB:

Glass Box Testing:
Process for evaluating the correctness or effectiveness of a piece of software while examining its algorithmic structure.

Testing Suite:
A software package designed to evaluate the correctness or effectiveness of another software solution.

Test Driven Design:
A software development process in which developers first create a test suite and then create the code to satisfy the test suite, e.g., Xtreme Programming.

Flow chart:
A graphic organizer that can be used to show the procedural pathways within a program.

CODE:

4.
```
def f(x):
    if int(x) == x:
        if x % 2 == 0:
            if x % 3 == 0:
                return "F is a multiple of 6"
            else:
                return "F is even"
        else:
            return "F is odd"
    else:
        return "F is not an integer"
```

PART 2

Note: In some Python versions, the raw_input command does not work. Instead, just use input.

VOCAB:

Unicode: Extended ASCII to include all world languages, including accent symbols.

~~Multi-type variable~~

Type Casting:
Converting data from one type to another, e.g., from string to int, potentially losing information.

Concatenation vs. Numeric Addition: When the + operator is between two strings, it concatenates, putting the second string of characters right after the first string of characters and into a single concatenated string. When the + operator is between two numbers, it performs numeric addition, resulting in an int or float.

Null String:
A string that contains no characters.

ANSWER:
7a. If the guess does not not equal the secret then the user guessed correct.

7b.
```
def guess_once():
 |  secret = random.randint(1, 4)
 |  print('I have a number between 1 and 4.')
 |  guess = int(input('Guess: '))
 |  if guess == secret:
 |   |  print('Right on! I was', str(guess) + '!\n')
 |  elif guess < secret:
 |   |  print('Too low - my number was', str(secret) + '!\n')
 |  elif guess > secret:
 |   |  print('Too high - my number was', str(secret) + '!\n')
```

CODE:
8.
```
def quizDecimal(low, high):
    n = input("Number? ")
    if str(n) != n:
        if n < low:
            return "No,", n, "is less then", str(low) + ".\n"
        elif n > high:
            return "No,", n, "is greater then", str(high) + ".\n"
        else:
            return 'Good!', low, '<', n, '<', high, "\n"
    else:
        return 'Numbers not letters!'
```

CONCLUSION:

1. Glassbox Testing evaluates the If statements and their structure to find bugs in the code.

2. Depends on how many you have and how many arguments are met.

3. A testing suite tests a program with predetermined conditions do see that it meets requirements. So that the programmer knows what the program needs to exactly do.