Alex Gutierrez

2a. The program that I created is a simple game of “Rock Paper Scissors Lizard Spock” and I got the idea for my program from “The Big Bang Theory.” The programming language that I used to create my program was Canopy Python and the purpose for my program is merely to play a game when one is bored. My program also has a minigame within it that only triggers when the player and the computer tie, resulting in the program presenting a simple coinflip to resolve it. In the video that was presented to you, I explained the abstraction and how it allowed the user to input one of six choices with an “elseif” command, and if the chose anything but the six choices, the program would respond with wiseass.

2b. The developmental process that went into making my program was simple, as all I needed to do was to make a way for the user to input certain commands and have my program respond to those commands randomly. The user would type in a command, and my program would choose a random randint from one to five and respond with the line of code that went with both the user’s command and the randint that the computer chose. There were hardly any difficulties in making my code, but I every time my program made a mistake such as always picking rock or displaying the wrong user input, I went back to the code that was repeating itself and altered it so that it would move on. The majority of the development for my program was independent, but one of my classmates did teach me how to create variables and incorporate them into my code, which in a sense was the basis for my entire program. The rest of the program however, was my own design, for although my classmate taught me how to incorporate variables, I was the one who implemented them into a game and a minigame.

2c.

```
#1
def rock(name):
    cpu = random.randint(1,5)
    if cpu == 1:
        print (name,' chose Rock and the CPU chose Rock. Tie!')
        cf(name)

    if cpu == 2:
        print ('The CPU chose Paper and' ,name, 'chose Rock. You lose!')

    if cpu == 3:
        print (name,' chose Rock and the CPU chose Scissors. You win!')

    if cpu == 4:
        print (name,' chose Rock and the CPU chose Lizard. You win!')

    if cpu == 5:
        print ('The CPU chose Spock and' ,name, 'chose Rock. You lose!')
#2
def cf(name):
        b = raw_input('\n\nHeads or Tails?:\n\n')
        p2 = b.lower()
        print('\n')
        if p2 == 'heads':
            heads(name)
        elif p2 == 'tails':
            tails(name)
        elif p2 == 'end':
            byebye(name)
            sys.exit()
```

My first algorithm is one of several nearly identical ones as the user has the choice of five different inputs for the game. In this algorithm, the user's choice is "rock" and the computer has the choice of picking one of five different responses based on what random integer it selected. In this instance, if the user chose to input "rock" and the computer selected the random integer 1, then the program would respond with the game resulting in a tie, and thus initiating the "cf" minigame. In the "cf" minigame, the code is again nearly identical as the one for "rpsls," the only difference being that the user only has two choices, heads or tails. The computer then has a much simpler job of selecting one of two different responses based on what the user inputed. If both the user and the computer chose heads, then the user won. If the user and the computer chose differently, then the player lost. Both of these algorithms are separate from each other and only connect when the first meets a certain condition, the tie. Although the second algorithm was not necessary, it adds to the overall experience of playing the game.

2d.

```
def choice(name):
    y = 0
    while y == 0:
        a = raw_input('\n\nRock, Paper, Scissors, Lizard, or Spock?:\n\n')
        p1 = a.lower()
        print('\n')

        if p1 == 'paper':
            paper(name)

        elif p1 == 'rock':
            rock(name)

        elif p1 == 'scissors':
            scissors(name)

        elif p1 == 'lizard':
            lizard(name)

        elif p1 == 'spock':
            spock(name)

        elif p1 == 'end':
            byebye(name)
            sys.exit()
```

My abstraction was necessary for my program to properly function because it is here where the user's input choices were first defined. If the input choices were never defined, then the subsequent algorithms would not even be able to be utilized. The abstraction defines six different user inputs, five inputs are for playing the game and the sixth input is for ending the game and exiting the program. Not only are the first five user inputs necessary to play the game, but without the final user input, the player would be stuck playing "rock paper scissors lizard spock" for an eternity.