Final Project Write-up
Megan Andree and Amelia McDowell

The goals of our final project were to create a fun game for kids to play to learn basic math skills by answering questions correctly in order to build a caterpillar.

The main function of our program first calls the instruction function. The instruction function prints out the instructions for the game in the terminal. The main function then prompts the user to select a level of difficulty for their math problems. While the variable to play again, playAgain, is equal to 'y':

The main function sets the accumulating variables, one for each body part that corresponds to a die, equal to zero. The main function also sets the boolean variables, builtCaterpillar and myWinOpen, equal to False. While the value for the variable of the number of incorrect answers, incorrect, is less than three AND the boolean variable representing that the entire caterpillar is complete, builtCaterpillar, is equal to False:

The main function calls the constructEquation1 function. The constructEquation1 function takes one parameter, choice, which is the user's choice of difficulty for the game. If the choice is equal to 1 or 2, the constructEquation1 function: generates two random numbers, numOne and numTwo, adds both of the numbers to a list called equation, adds an 'x' variable to the list, generates a sign, + or - , randomly if the choice was 1 and adds it to the list, or adds a multiplication sign to the list if the choice was 2. If the user's choice was not 1 or 2, the constructEquation1 function prints out that the user did not enter a valid selection from the menu. The constructEquation1 function returns equation, the list, to the main function. The main function then assigns the returned list to a variable called equation.

The main function then calls constructEquation2. constructEquation2 takes one parameter, equation. The constructEquation2 function appends the string of each item in the list, equation, to a new list, equation2. If the second item in equation2 is equal to a + or – sign, the constructEquation2 function will randomly generate a mathematical equation (addition or subtraction) in the form of a string by accumulating a string based on random values that correspond to items in the list. If the second item in equation2 is equal to a multiplication sign (X), the constructEquation2 function will check to see which of the numbers (items 1 and 2 of the list) are greater. The constructEquation2 function will see if the integer division of these numbers is equal to the regular (decimal number division). If they are equal, the function will put the x variable on the left side of the = sign in the equation. If it is not equal, the function will put the x variable on the right side of the = sign, because one number is not a multiple of the other number. The constructEquation2 function will generate a multiplication equation in the form of a string. If the two numbers (items 1 and 2 of the list) are equal, the constructEquation2 function will put the x variable on the right side of the equation. The constructEquation2 function returns the string of the mathematical equation called string. The main function assigns the return string to a variable called equationString.

The main function will then print out the generated mathematical equation and prompts the user for their answer. The main function then calls the solve function. The solve function has one parameter, equationString. The solve function splits the string of the equation and assigns it to a list called splitString. If splitString includes a – or + sign, the splitString function calls the sameFormat function. The sameFormat function takes one parameter, equationString, and returns the string of the equation without uppercase letters or spaces. The solve function will solve the equation for the value of x depending on where x is in the function and depending on if the equation is subtraction or addition. The solve function will return the value of x. If splitString includes a X for multiplication, the solve function will solve the multiplication problem for x, depending on the position of x within splitString.

The solve function will return the value of x to the main function, and the main function will assign the value of x to a variable called actualAnswer.

If actualAnswer is equal to the user's answer, The roll variable creates a random integer that is presented to the user in the terminal. If the roll is equal to one and the user does not have a body, the program opens an image window(myWin). The variable bodyCount calls the buildBody function that takes the parameters of roll, bodyCount, and myWin. This function places the image of a body inside the window and returns the bodyCount variable after accumulating it, which tells the program that the user already has a body. If the roll is equal to two, and the bodyCount is equal to one, the headCount variable calls the buildHead function with parameters roll, bodyCount, headCount, and myWin. This function places the image of a head inside the window and returns the headCount variable after accumulating it, which tells the program that the user already has a head. If the roll is equal to three, the bodyCount is equal to one, and the headCount is equal to one, the hatCount variable calls the buildHat function with parameters roll, headCount, hatCount, and myWin. This function asks the user if they would like a bow, hat, or antennae on their caterpillar and then places the image of their choice on the head of the caterpillar inside the window. The function returns the hatCount variable after accumulating it, which tells the program that the user already has a hat. If the roll is equal to four, and the bodyCount is equal to one, and the headCount is equal to one, the eyeCount variable calls the buildEye function with parameters roll, bodyCount, headCount, and myWin. This function places the image of eyes on the caterpillar inside the window and returns the eyeCount variable after accumulating it, which tells the program that the user already has eyes. If the roll is equal to five, the bodyCount is equal to one, and the headCount is equal to one, the teethCount variable calls the buildTeeth function with parameters roll, headCount, teethCount, and myWin. This function asks the user if they would like teeth, lips, or a tongue on their caterpillar and then places the image of their choice on the head of the caterpillar inside the window. The function returns the teethCount variable after accumulating it, which tells the program that the user already has a mouth. Next, the builtCaterpillar calls the checkIfWin function, which takes the parameters of bodyCount, headCount, hatCount, eyeCount, and teethCount. This function checks if each of these variables is equal to one and returns True if that is correct. If the user does not answer the question correctly, they do not get to roll and the incorrect variable accumulates and prints out the amount of strikes that they have. If they get three questions wrong, they lose and the program prints out the final score. If the builtCaterpillar function is True, the program prints that the user has won and prints their final score and closes the image window.

The user will be asked if they would like to play again. If the user enters y, the while loop boolean variable, playAgain, is still equal to y and the game will continue for a new round. If the user enters n, the while loop boolean variable playAgain will not be equal to n, and the game will end and will return to the terminal line for the folder.

To play our game, the user first chooses a level of difficulty: 1 for addition and subtraction problems, 2 for single digit multiplication problems. Th user then types the answers to the questions provided directly in the terminal. If the user answers correctly, they get to roll a die. Each side of the die corresponds to a different part of a caterpillar that they can use to build a whole caterpillar. However, they must roll a body before they can build any other part of the caterpillar and they must roll a head second before they can add the pieces that belong on the head. When the caterpillar has 5 parts, they win the game. If they answer three questions incorrectly, the program will tell them that they lose and ask if they want to play again.

The program should be run as:
python3 finalProject.py

We tested the mathematical equation part of the code separately, and in pieces as we went. We first wrote the code for the constructing of the equation, and tested that code. We then wrote the code for solving the equation and tested that code as well. We then created the if statements for the images of the caterpillar and tested that code individually from the mathematical equation section. We put the two halves of the code together and tested it completed. For the future, we would break the code up into more functions so it is easier to understand and modify. We would also add the print statements from the instruction function to a file, and just read the file to print the instructions.