Design of Lab9– Jacob Bollinger

Problem:

Design software to play a game called Flip on the console.

Plan:

Start by importing randint from random.

Define a function named initializeLists(). Initialize rDice, mDice, and hDice to blank lists. Initialize rForbiddenDice and hForbiddenDice to list of [False] \* 5. Create a for loop to for i in range(5). Inside the for loop, append a random integer between 1 and 6 to rDice and hDice. Outside the for loop, sort rDice, mDice, and hDice. Return rDice, mDice, hDice, rForbiddenDice, and hForbiddenDice.

Define a function named displayLists() witht the parameters robotList, middleList, and humanList. This function will print 3 lines with the first being ‘Robot List:’ and robotList, second being ‘Middle List:’ and middleList, and the list being ‘Human List’ and humanList.

Define a function moveHuman() with the parameters r, m, and h. Call the function displayLists with the parameters r, m, and h. Set a variable move equal to the keyboard in input with the prompt “Please enter a move with fD or tD where D is a digit”. Return move.

Define a function called moveRobot() with the parameters r, m, and h. Set the list moveChoice equal to [‘f’, ‘t’] and digitChoice to [1, 2, 3, 4, 5]. Redefine digitChoice to digitChoice[0:len(h)]. Set the variables move equal to sample(moveChoice, 1) and digit equal to sample(digitChoice, 1). Return move and digit.

Define a function humanTurn with the parameters move, r, m, h, and forbiddenList. Create an if statement to test if move[0] is equivalent to ‘f’. If so call flip with parameters move[1], h, and forbiddenList. Create an else if move[0] is equivalent to ‘t’. If so set forbiddenList equal to the call trash with the parameters move[1], r, m, h, True, and forbiddenList. Return forbiddenList.

Define a function robotTurn with the parameters move, r, m, h, and forbiddenList. Create an if statement to test if move[0] is equivalent to ‘f’. If so call flip with parameters move[1], h, and forbiddenList. Create an else if move[0] is equivalent to ‘t’. If so set forbiddenList equal to the call trash with the parameters move[1], r, m, h, True, and forbiddenList. Return forbiddenList.

Define a function named trash() with the parameters digit, r, m, p, isHuman, and forbiddenList. Set the string digit equal the the integer version of digit. Create an if statement to test if isHuman is equal to true. If so append(r[digit – 1]) to the list m and remove(r[digit – 1]) from the list r. Create another if statement to test if the length of list m is greater than 2 and if m[0] + m[1] is less than or equal to m[-1]. If so, append m[0] to the list h, append m[1] to the list h, and remove m[0] and m[1] from the list m. Create an else statement for the first if statement. Inside, call the function displayLists(). Append(h[digit – 1]) to the list m and remove(h[digit – 1]) from the list h. Create another if statement to test if the length of list m is greater than 2 and if m[0] + m[1] is less than or equal to m[-1]. If so, append m[0] to the list r, append m[1] to the list r, and remove m[0] and m[1] from the list m. Outside both if statements sort the lists r, m, and p. Set forbiddenList equal to [False] \* 5. Return forbiddenList.

Define a function named flip() with the parameters digit, correctList, and forbiddenList. Set the string digit equal to the integer version of digit. Create an if statement to test if forbiddenList[digit – 1] is equivalent to False. If so, set correctList[digit – 1] equal to 7 – correctList[digit – 1]. Create an else statement to print “You have already flipped this die.”

Define a final function named main. Start it by initializing the integers rWins and hWins. Set the lists robotList, middleList, humanList, robotForbiddenList, and humanForbiddenList equal to initializeLists(). Set a while loop to run while the length of humanList and robotList is not equal to 0. Set the variable move equal to moveHuman(robotList, middleList, and humanList). Set humanForbiddenList equal to humanTurn(move, robotList, middleList, humanList, and humanForbiddenList). Create an if statement to test if robotList is equal to []. If so, print “player wins” and pWins equal to itself plus 2. Create an else statement and inside set move equal to moveRobot(robotList, middleList, humanList). Call the function robotTurn with the parameters move, robotList, middleList, and humanList.

Outside all of the functions, call main().