3. WRITTEN RESPONSES

3 a.

3.a.i.

The overall purpose of this program is to entertain the audience with a clicking game.

3.a.ii.

The functionality of the program is that it allows the user to gain points and beat other players by clicking on the fruits as they appear randomly as time decreases, staying aware of the dangerous grape that will cost them a life if clicked.

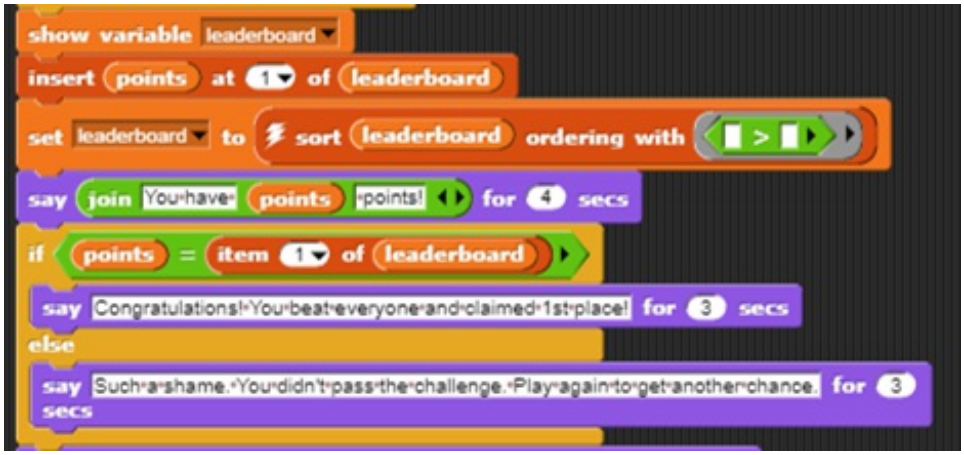
3.a.iii.

The input to the program is the user-generated answer to the question of which level the user wishes to play at, easy or hard. The output is that the speed at which the fruits move is changed based on the level, with the speed being slower on easy and faster on hard.

3 b.

3.b.i.

3.b.ii.



3.b.iii.

The name of the list used in the program is ‘leaderboard’.

3.b.iv.

The data in the list ‘leaderboard’ represents the scores each user obtained through the game in descending order.

3.b.v.

The list ‘leaderboard’ manages complexity by holding the score values in an easily accessible and codable place. It would be difficult to order the points from greatest to least without the ‘leaderboard’ list. If I did not use a leaderboard to organize the points, I would need to manually write out each score onto the screen and use if-else statements to see if each score is greater than or less than the other values. This would order it from greatest to least but in a more troublesome way than if I used a leaderboard.

3 c.

3.c.i.

3.c.ii.

3.c.iii.

The student-developed procedure ‘level’ allows for different speeds on the sprites depending on the user-inputted variable of ‘answer’. If the user inputs easy, the sprites will go to random positions and wait three seconds before moving. If the user inputs hard, the sprites will go to random positions but will wait one second before moving.

3.c.iv.

In the procedure ‘level’, there is an if else statement with the condition that if the parameter ‘#’, which is changed to the variable 'answer', is equal to easy, the sprite will go to a random position and then wait three seconds. This will be repeated until the variable 'time' equals zero. If the above condition is false, the sprite will go to a random position and wait one second, being repeated until the variable ‘time’ is equal to zero.

3 d.

3.d.i.

First call:

The first call is easy difficulty.

Second call:

The second call is hard difficulty.

3 d.ii.

Condition(s) tested by first call:

If the user inputs ‘easy’, then the procedure will run the ‘if’ part of the if else statement and skip the 'else' portion.

Condition(s) tested by second call:

If the user inputs ‘hard’, then the procedure will skip the ‘if’ part of the if else statement and run the ‘else’ portion.

3.d.iii.

Results of the first call:

The first call will result in the sprites waiting for three seconds before going to a random position on the stage.

Results of the second call:

The second call will result in the sprites waiting for one second before going to a random position on the stage.