3. WRITTEN RESPONSES

3 a.

3.a.i.

The overall purpose of the program is to lessen boredom by simulating a timed fishing game that tracks score.

3.a.ii.

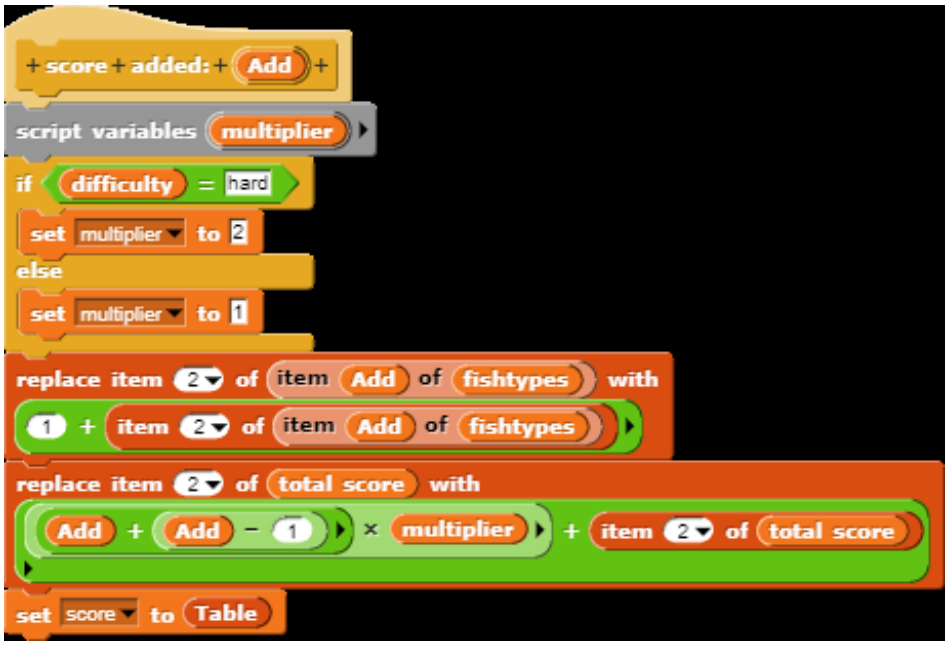
The video demonstrates fishing gameplay with the boat movement, fishing gameplay with the hook that goes down and back to the boat to catch fish, scorekeeping with the total score calculated from fish caught in time, and scorekeeping by tracking amounts of each fish caught in time.

3.a.iii.

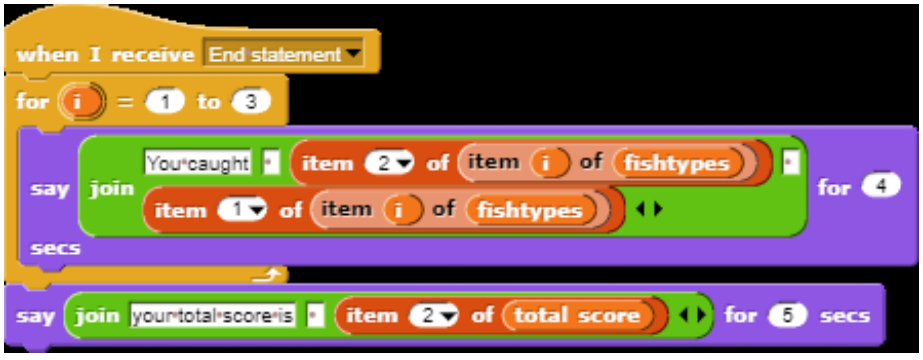
In the video, the player inputs by pressing the ‘a’ and ‘d’ keys, and the output for these keys are the boat's left and right movement. Also, the player inputs by pressing the spacebar, and the program outputs a hook sprite that goes underneath the boat and moves downward and returns to the boat so it can “catch” or touch other sprites, affecting different parts of the game. When the time ends, the boat will output a message of the amounts of each fish caught and the total score.

3 b.

3.b.i.



3.b.ii.



3.b.iii.

The list being used in the code above is “fishtypes,” and it tracks the amount of fish caught. In the second image, the list is being used to tell the player the amount of each type of fish caught.

3.b.iv.

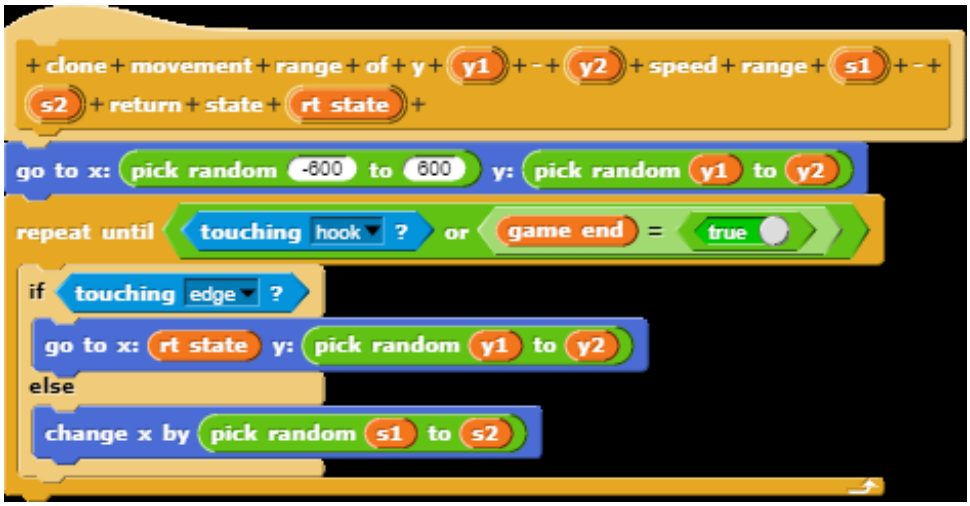
The data in this list represents the type of fish and the number of a specific fish caught.

3.b.v.

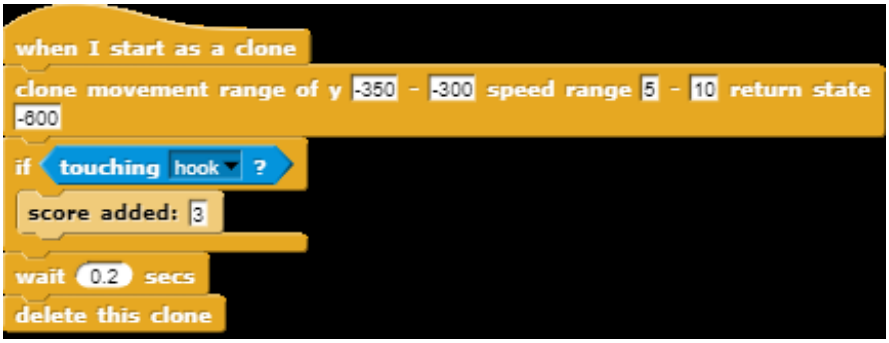
The list manages complexity by keeping the data altogether in one easy list. Without the list, the program would need to have multiple variables for each type of fish and their own amounts, making the program unnecessarily longer. The list compiles all that fish data into one spot. Since all the data of the fish types and their respective amounts are in this one list, it is easier to use the information since the info can be extracted easier. If another fish was added to the fishing game, one could simply add data of the fish into the list instead of making a new variable.

3 c.

3.c.i.



3.c.ii.



3.c.iii.

The procedure “clone movement” determines where a clone of a sprite spawns, how far it can move, and where it goes when it reaches the end of the screen. This contributes to the overall program by making it more challenging to catch a fish sprite with the hook because the procedure causes each fish clone to spawn and move randomly at a range of speeds or height.

3.c.iv.

The procedure starts by moving a newly spawned clone to a random x position within the screen and a random height within a range specified by the “y1” and “y2” parameters.

Then, a loop begins until either the game ends or the clone touches the hook.

Inside the loop, is an if-else statement where if the clone touches the edge of the right side of the screen, the clone would teleport left to a position specified by the “rt state”(return state) and to random height within a range specified by “y1” and “y2” parameters.

If the clone is not touching the right side of the screen, the clone would move right a random number steps within the ranges specified by the “s1” and “s2” parameters.

3 d.

3.d.i.

First call:

The first call is a returning call.
“go to x:(rt state) y: [pick random(y1) to (y2)]”

Second call:

The second call is a normal movement call moving right.
“change x by [pick random(s1) to (s2)] ”

3 d.ii.

Condition(s) tested by first call:

The first call tests whether the clone sprite is touching the edge of the right side of the screen.

Condition(s) tested by second call:

The second call tests whether the clone sprite is not touching the edge of the right side.

3.d.iii.

Results of the first call:

The cloned sprite moves left to the amount specified by the “rt state” parameter and to a random height within the ranges “y1” to “y2.” This will make the clone continue to move without leaving the screen.

Results of the second call:

The cloned sprite moves right a random amount within a range at or between the “s1” and “s2” parameters.