2d. Capture and paste a program code segment that contains an abstraction you developed individually on your own (marked with a rectangle in section 3 below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (*Must not exceed 200 words*)

<table>
<thead>
<tr>
<th>Row and Task</th>
<th>Decision Rules</th>
</tr>
</thead>
</table>
| **Row 7 - Code Segment in Response 2D**   | - Responses that use existing abstractions to create a new abstraction, such as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.  
  **Do NOT award a point if any one of the following is true:**  
  - the response is an existing abstraction such as variables, existing control structures, event handlers, APIs; or  
  - the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or  
  - the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction). |
| **Row 8 - Response 2D**                   | - Responses should not be penalized for explanations of abstractions that are not developed by the student.  
  **Do NOT award a point if any one of the following is true:**  
  - the explanation does not apply to the selected abstraction; or  
  - the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction). |
**Student Response A - [Video] [Written Response]**

```
world.setrandomcolor ([Obj] chicken)
randomNum = 1
  // Gives all the chickens different colors
  randomNum set value to (random number minimum = 1 maximum = 9 integerOnly = true)
  If (randomNum == 1)
    chicken set color to duration = 0.5 seconds
  Else
    If (randomNum == 2)
      chicken set color to duration = 0.5 seconds
    Else
      If (randomNum == 3)
        chicken set color to duration = 0.5 seconds
      Else
        If (randomNum == 4)
          chicken set color to duration = 0.5 seconds
        Else
          If (randomNum == 5)
            chicken set color to duration = 0.5 seconds
          Else
            If (randomNum == 6)
              chicken set color to duration = 0.5 seconds
            Else
              If (randomNum == 7)
                chicken set color to duration = 0.5 seconds
              Else
                If (randomNum == 8)
                  chicken set color to duration = 0.5 seconds
                Else
                  If (randomNum == 9)
                    chicken set color to duration = 0.5 seconds
                  Else
                    Do Nothing
```

The abstraction I chose is my method “setrandomcolor.” This method uses math to set a random number to each of the chickens that are used in my game. The method then uses logical if/else statements to assign different colors to the range of numbers that are given to the chickens. This abstraction helped manage the complexity of my program because it ensured that I would not have to put together the large amount of code every time that I called the method. I call this method once in the beginning of my program and then later if there are not any blue chickens left. Because of this abstraction I did not have to re-write the same ten if/else statements more than once.

**Row 7**

The response earned a point for this row. The boxed code segment in the response represents a valid abstraction (a procedure or method).

**Row 8**

The response earned a point for this row. The response explains how the abstraction manages complexity: "I did not have to re-write the same ten if/else statements more than once" and "it ensured that I would not have to put together the large amount of code every time that I called the method."

---

**Student Response B - [Video] [Written Response]**

```
// print(float x)
for float x in range(0, 10.0)
paint(x) = x * 10

// For a [Pencil] you can draw a line or a dot.
// For a [Stamps] you do not.
if [Stamps] == true

// The code given in the response represents an abstraction (a procedure).
```

The response earned a point for this row. The code given in the response represents an abstraction (a procedure).

**Row 8**

The response earned a point for this row. The response explains how this abstraction manages complexity by "improving readability ... and making debugging easier."
An abstraction developed individually is quiz(), which is used in the function run_quiz(). quiz() quizzes the user, records terms the user messes up on, calculates total points_scored and streak-points, and gives the user motivational messages based on their streak-points. Example of using a mathematical concept is using modulus-2 to determine the parity of the user’s streak-points. Example of a logical concept is an if-else statement that determines if the user answered correctly. My function does this by returning a Boolean value (True or False) which describes whether the user’s answer equals correct answer, and using that value to determine whether or not to run add 1 each to points_scored and streak-points. Having this abstraction helped manage the complexity of this program by one, improving readability, and two, making debugging easier. Firstly, since all the code regarding the quiz is in a function, running the quiz itself only requires quiz(). Second, this abstraction made debugging significantly easier. An example of this is when I found out that the quiz was not functioning the way I had expected it to, so I immediately started looking for errors in the quiz() function rather than having to look through my entire program for the error.

Student Response C - [Video] [Written Response]

Row 7
The response earned a point for this row. The selected code segment shows the use of a list (cardList) to create a new abstraction to represent a collection of cards.

Row 8
The response earned a point for this row. The response explains how the abstraction manages program complexity: "the iterative process of going through each item in the list increases code efficiency because I no longer had to add every card to the background separately ..." Use of the list helps manage complexity by allowing for the use of a loop to process the list, reducing the amount of coding as explained: "I had around 9 separate addObject statements and 9 separate calls to the card's act() method."
and reset the x coordinate variable to 105, Create Sample C 2 of 3 which sets the image to the far left. Using those updated variables, the program then adds the card object.

Student Response D - [Artifact] [Written Response]

While I was writing the program for my bowling game, I realized something. Every time the bowling ball touched a bowling pin, I programmed it to hide, go back to its original position, set the size to the original 10%, and then be visible again, I noticed that this code could be made shorter and simpler. I figured that if I made a custom block called “position”, I could fit all of those blocks of code into “position” and make it an abstraction block. This helped me significantly because it ensured that every time I would look at the code, I wouldn’t get confused to see the same things repeating over and over again and it would make the code seem simpler and more human friendly. It also helped manage the complexity of my program because every time I looked at the block of code “position”, it made me understand that the block was simply asking for the bowling ball to be returned back to its original position so that the user could roll the ball again.

Scoring Guidelines

Row 7
The response earned a point for this row. The block given is an abstraction (a procedure).

Row 8
The response earned a point for this row. The response explains how complexity is managed: "I noticed this code could be made shorter and simpler ... it ensured that every time I would look at the code, I wouldn’t get confused to see the same things repeated over and over."

Student Response E - [Video] [Written Response]

This particular abstraction is used to determine if the word you type in, is correct. This abstraction uses mathematical concepts by determining if the word you type in and the actual term, are equal. If they do happen to be equal, then this will be shown in the "lblWriteWrong." Furthermore, this abstraction uses logical concepts by determining if the word the user types in is true, then it will be displayed as correct through the Write Wrong label. However if the word the user types in is a false word, then it will show that

Scoring Guidelines

Row 7
The response earned a point for this row. The supplied code block represents a student-developed abstraction, which is a procedure.

Row 8
The response earned a point for this row. The response gives a reason why the abstraction manages complexity: "By creating this abstraction it makes the general coding clearer and easier to read as it is already being used once."
it is incorrect through the Write Wrong label. By creating this abstraction it makes the general coding clearer and easier to read as it is already being used once.

**Student Response F - [Video] [Written Response]**

```javascript
function Time() {
    myInterval = setInterval(function() {
        seconds = seconds - 1;
        setText("Time",seconds);
        if (seconds === 0) {
            if (Score > HighScore) {
                HighScore = Score;
                setValue("HighScore", Score, function () {
                    setText("HighScore", HighScore);
                });
            }
            setText("txtScore","You have received a score of " + Score + " Nyan Cats that you clicked!!");
            setScreen("GameOverScreen");
            clearInterval(myInterval);
        }
    }, 1000);
}
```

In order to use the , such as certain lines of code are not repeated, I made my code use abstraction is by using this Time() function, and implementing it into the event handlers so the code is reduced and not being repeated on certain event handlers. My first thought for my Time() function was using the setInterval() function and the if statement and applying it to certain event handlers over again, such that the 30 second interval would run, and the screen would change when the 30 seconds would end. Using abstraction helped me implement the setInterval() function and the if statement such that it would be unnecessary to repeat code and have extra lines of code, adding less complexity to the algorithms that I used into making this code.

**Scoring Guidelines**

<table>
<thead>
<tr>
<th>Row</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

The response earned a point for this row. The code segment given represents an abstraction (a procedure or function).

<table>
<thead>
<tr>
<th>Row</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>1</td>
</tr>
</tbody>
</table>

The response earned a point for this row. The response explains how the abstraction manages complexity for the program by stating: "the code is reduced and not being repeated on certain event handlers."

**Student Response G - [Video] [Written Response]**

The purpose of this abstraction is to cause the snake to grow every time it eats an apple. This is done by ceating a new clone that will follow the snake every time it eats an apple. This abstraction also increases the speed of the snake each time it collects an apple. This causes it to be more challenging to get the apple and also avoid the edges.

**Scoring Guidelines**

<table>
<thead>
<tr>
<th>Row</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>1</td>
</tr>
</tbody>
</table>

The response earned a point for this row. The boxed code fragment does not represent an abstraction.

<table>
<thead>
<tr>
<th>Row</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0</td>
</tr>
</tbody>
</table>

The response DID NOT earn a point for this row. The response does not explain how the boxed code manages complexity in the program. Making the program more challenging is not the purpose of an abstraction.
### Student Response H - [Video] [Written Response]

```javascript
//
// if wrong answer is clicked -- leads back to the beginning
onEvent("click", function(event) {
  setScreen("Q1");
});

The purpose of my program is to educate and encourage people to learn foreign languages. To do that, my game/quiz asks questions, and the user must select one of four answers, one is in Spanish, one is in French, one is in Japanese, and one is in German. Because repetition is key for learning a new language, I decided to incorporate it into my program. Except for three questions where the user types in a response, the rest are multiple choice questions. For each question, there is only one correct answer, so when any one of the incorrect answers is clicked, the game over screen comes up. On that screen the user then clicks on a button to try again. The trick of the game is that every time an incorrect answer is clicked, it brings you back to the first question. This program code segment allows the game to work as a constant loop, always waiting for the game over screen to appear.
```

<table>
<thead>
<tr>
<th>Row 7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>The response DID NOT earn a point for this row. The code fragment given is not a student-developed abstraction. The abstraction given, an event handler, is built in to this language.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row 8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>The response DID NOT earn a point for this row. The response does not explain how the supplied abstraction manages complexity for the program. The response only explains how the code fragment works in relation to the entire program.</td>
<td></td>
</tr>
</tbody>
</table>

### Student Response I - [Video] [Written Response]

The blue rectangle in the image showcases a code segment which implements a mathematical function to the program which was developed independently. The code takes the value given to countdown by the user in the dropdown textbox or code getText from line 4 and subtracts one from it. After so, the code segment displays the result in the countdown slot on the screen. This code segment is part of a function which serves as an algorithm to repeat the code segment. Without the code segment the hole program would not run as there would be no segment giving it the means to subtract and reach zero, therefore the displayed value would always be the one input by the user.

<table>
<thead>
<tr>
<th>Row 7</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>The response DID NOT earn a point for this row. The boxed code fragment does not represent an abstraction.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row 8</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>The response earned a point for this row. The response does not explain how the boxed code manages complexity in the program. The response merely states how the code fragment works.</td>
<td></td>
</tr>
</tbody>
</table>
### Student Response J - [Video] [Written Response]

<table>
<thead>
<tr>
<th>Row</th>
<th>Code Segment</th>
<th>Scoring Guidelines</th>
</tr>
</thead>
</table>
| 7   | ```
31 onEvent(\`\`gobackbutton2\`, \`\`click\`, function(event) {
32 console.log(\`\`gobackbutton2_clicked\`;)
33 }
34 onEvent(\`\`button3\`, \`\`click\`, function(event) {
35 setScreen(\`\`HelloScreen\`;);
36 }
37 onEvent(\`\`button4\`, \`\`click\`, function(event) {
38 setScreen(\`\`WelcomeScreen\`;);
39 }
40 onEvent(\`\`button5\`, \`\`click\`, function(event) {
41 setScreen(\`\`WelcomeScreen\`;);
42 }
43 onEvent(\`\`button6\`, \`\`click\`, function(event) {
44 setScreen(\`\`WelcomeScreen\`;);
45 }
46 onEvent(\`\`button7\`, \`\`click\`, function(event) {
47 setScreen(\`\`HelloScreen\`;);
48 }
49 onEvent(\`\`button8\`, \`\`click\`, function(event) {
50 setScreen(\`\`WelcomeScreen\`;);
51 }
52 onEvent(\`\`button9\`, \`\`click\`, function(event) {
53 setScreen(\`\`WelcomeScreen\`;);
54 }
55 onEvent(\`\`button10\`, \`\`click\`, function(event) {
56 setScreen(\`\`WelcomeScreen\`;);
57 }
58 ``` |
|     | The response **DID NOT earn a point for this row**. The code segment given does not represent a student-defined abstraction. An event handler is built-in to this language. |
| 8   | ```
59 onEvent(\`\`button11\`, \`\`click\`, function(event) {
60 setScreen(\`\`WelcomeScreen\`;);
61 }
62 onEvent(\`\`button12\`, \`\`click\`, function(event) {
63 setScreen(\`\`WelcomeScreen\`;);
64 ``` |
|     | The response **DID NOT earn a point for this row**. The response does not explain how this abstraction chosen manages complexity of the program. The response states *"It was complex making the game move automatically to the next question after you answer it correctly."* This is not the correct meaning of complexity for this row. |

This manages the app so you get where you want to be questioned on. Allowing you to go to the home screen, after you finish your question. The game has a go back button which allows you to go home after you finish. But *it was complex making the game move automatically to the next question after you answer it correctly.* This was great experience making this computational artifact.