Create PT - Written Response

Video Submit one video in .mp4, .wmv, .avi, or .mov format that demonstrates the running of at least one significant feature of your program. Your video must not exceed 1 minute in length and must not exceed 30MB in size

Prompt 2a. Provide a written response or audio narration in your video that:

- identifies the programming language;
- identifies the purpose of your program; and
- explains what the video illustrates.

(Must not exceed 150 words)

The program language that I used is app lab in code.org to write my program. The purpose of my program is to create something fun that can interact with the player as well as provide a challenge to the user. The video illustrates how score is being updated each time the apple is caught in the basket and also shows how the game functions.

2b. Describe the incremental and iterative development process of your program, focusing on two distinct points in that process. Describe the difficulties and / or opportunities you encountered and how they were resolved or incorporated. In your description clearly indicate whether the development described was collaborative or independent. At least one of these points must refer to independent program development. *(Must not exceed 200 words)*

I wrote program firstly by giving movement to the basket using arrow keys as well as allowing the apples to drop from the sky to be caught by the basket which would then either give a point for a score if caught or subtract a point for lives if missed. I developed this code independently. When I first started my program I found difficulty in which the basket would go off the screen if pressing the arrow buttons excessively. To resolve this problem I created another function that restricted the zones on the x axis to make sure the basket didn't go off the screen. I encountered another problem when testing in which I tried to get the apple to reset once it interacted with the basket. To resolve this I created a if statement when the apple and baskets position reached the same point, I reset the position to the top of the screen again.

2c. Capture and paste a program code segment that implements an algorithm (marked with an **oval** in **section 3**) and that is fundamental for your program to achieve its intended purpose. This code segment must be an algorithm you developed individually on your own, must include two or more algorithms, and must integrate mathematical and/or logical concepts. Describe how each algorithm within your selected algorithm functions independently, as well as in combination with others, to form a new algorithm that helps to achieve the intended purpose of the program. *(Must not exceed 200 words)*

Code Segment

```
var BasketX = getProperty("Basket","x");
var BasketY = getProperty("Basket","y");
var speed = 2;
var score = 0;
var lives = 3;
onEvent("StartButton", "click", function( ) {
  setScreen("GameScreen");
  setPosition("Apple1", 140, 0, 35, 30);
  Game();
});
function Game() {
  MoveBasket();
  moveApple();
}
function MoveBasket() {
  onEvent("GameScreen", "keydown", function(event) {
    if (event.keyCode == '39'&& getXPosition("Basket")<255) {
      setPosition("Basket", getXPosition("Basket")+15, getYPosition("Basket"));
    } else if ((event.keyCode == '37'&& getXPosition("Basket")>0)) {
      setPosition("Basket", getXPosition("Basket") - 15, getYPosition("Basket"));
    }
    }
  );
}
function moveApple() {
  var loop = timedLoop(1, function() {
    setProperty("Apple1", "y", getYPosition("Apple1")+speed);
    setPosition("Apple1", getXPosition("Apple1"), getYPosition("Apple1"), 30, 30);
    checkCollision();
  });
}
function checkCollision() {
  if (getXPosition("Apple1")>getXPosition("Basket")&&getXPosition("Apple1")<getXPosition("Basket")+35&&getYPosition("Apple1")<380&&getYPosition("Apple1")>370) {
    score++;
    setPosition("Apple1", randomNumber(0, 290), 0, 30, 30);
  } else if ((getYPosition("Apple1"))>=450) {
    lives--;
    setPosition("Apple1", randomNumber(0, 290), 0, 30, 30);
  }
  setText("Lives", lives);
  setText("Score", score);
  if (lives==0) {
    stopTimedLoop();
    setScreen("LoseScreen");
    lives = 3;
    score = 0;
  }
}
onEvent("TryAgainButton", "click", function( ) {
  setScreen("StartScreen");
});
```

Written Response

My function Game calls both MoveBasket and moveApple which are both student developed functions. Game makes sure that the other functions are both functioning. MoveBasket utilizes mathematical as well as logical concepts to identify the position of the basket and move it using arrow keys. It also set the basket's position and makes sure the basket is restricted inside of the screen so it doesn't go off of it. MoveApple uses a loop to set the speed of the falling apples as well as call another function known as CheckCollision which resets the position of the apple once it reaches the minimum height also using mathematical concepts and if statements. Both algorithms are essential because without the first algorithm the basket would be static and would go off the screen and without the second algorithm the apple would be static therefore defeating the whole purpose of the game which is to catch the apples with the moving basket.

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**). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. *(Must not exceed 200 words)*

Code Segment

```
var BasketX = getProperty("Basket","x");
var BasketY = getProperty("Basket","y");
var speed = 2;
var score = 0;
var lives = 3;
onEvent("StartButton", "click", function( ) {
  setScreen("GameScreen");
  setPosition("Apple1", 140, 0, 35, 30);
  Game();
});
function Game() {
  MoveBasket();
  moveApple();
}
function MoveBasket() {
  onEvent("GameScreen", "keydown", function(event) {
    if (event.keyCode == '39'&& getXPosition("Basket")<255) {
      setPosition("Basket", getXPosition("Basket")+15, getYPosition("Basket"));
    } else if ((event.keyCode == '37'&& getXPosition("Basket")>0)) {
      setPosition("Basket", getXPosition("Basket") - 15, getYPosition("Basket"));
    }
    }
  );
}
//Algorithm 2 and my Abstraction
function moveApple() {
  var loop = timedLoop(1, function() {
    setProperty("Apple1", "y", getYPosition("Apple1")+speed);
    setPosition("Apple1", getXPosition("Apple1"), getYPosition("Apple1"), 30, 30);
    checkCollision();
  });
}
function checkCollision() {
  if (getXPosition("Apple1")>getXPosition("Basket")&&getXPosition("Apple1")<getXPosition("Basket")+35&&getYPosition("Apple1")<380&&getYPosition("Apple1")>370) {
    score++;
    setPosition("Apple1", randomNumber(0, 290), 0, 30, 30);
  } else if ((getYPosition("Apple1"))>=450) {
    lives--;
    setPosition("Apple1", randomNumber(0, 290), 0, 30, 30);
  }
  setText("Lives", lives);
  setText("Score", score);
  if (lives==0) {
    stopTimedLoop();
    setScreen("LoseScreen");
    lives = 3;
    score = 0;
  }
}
onEvent("TryAgainButton", "click", function( ) {
  setScreen("StartScreen");
```

Written Response

My code utilizes abstraction by involving a function to repeat itself several times to manage complexity and make the code easier to understand. The function moveApple calls CheckCollision each time the apple reaches a certain location in the game. This allows the code to check the position of the apple and using mathematical and logical statements, reset the apple to its intended position either adding a point to the score or subtracting a life. This abstraction makes my program more manageable by taking a part of the code that would have repeated itself multiple times and condensing it to reduce complexity.