# Create – Applications From Ideas
# Written Response Submission Template

Please see Assessment Overview and Performance Task Directions for Student for the task directions and recommended word counts.

**Program Purpose and Development**

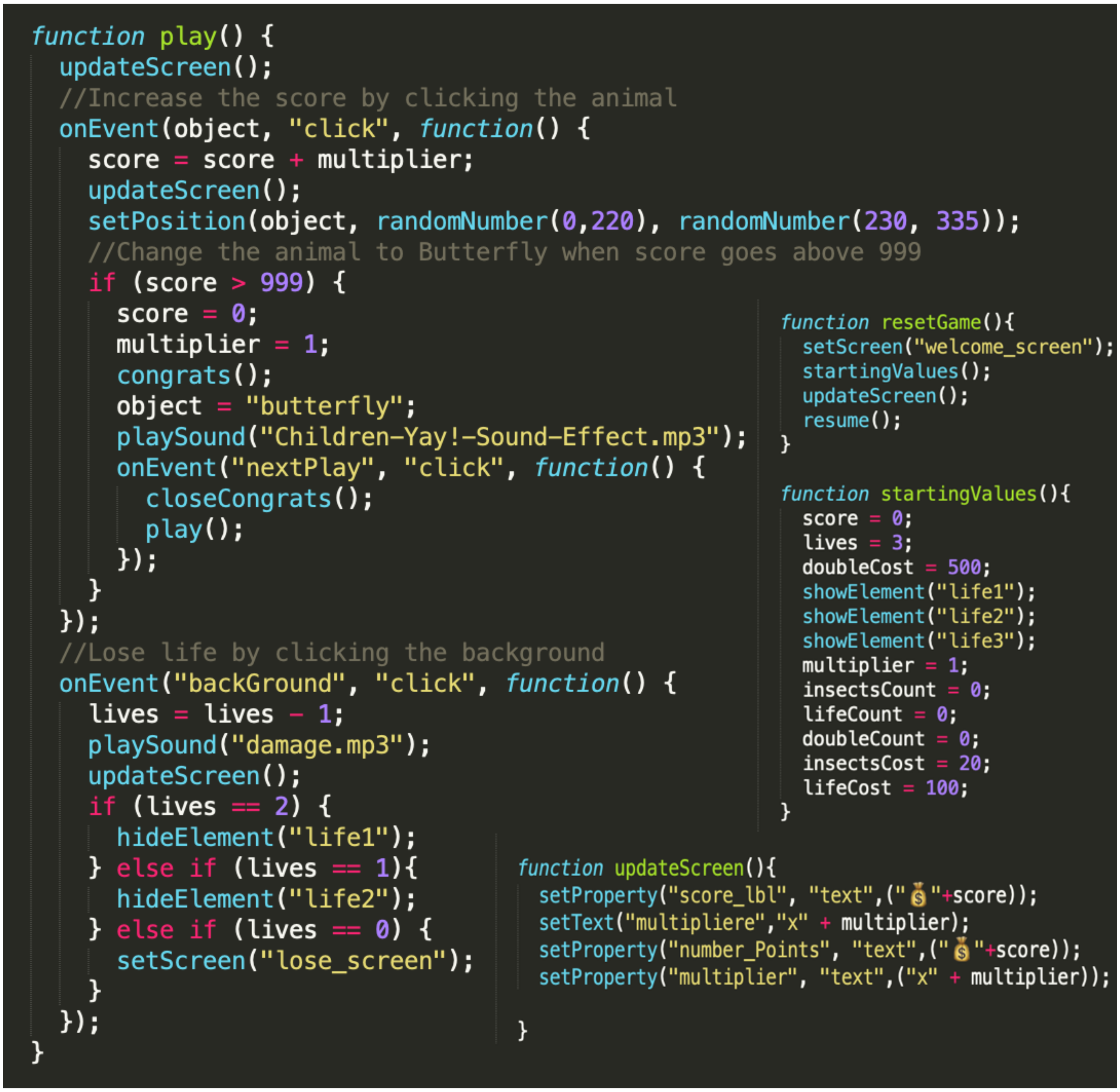
2a)

My program, 'FrogFly,' is a game created with JavaScript in the Code.org App Lab. The purpose of this program is to be a fun clicker-game for users to play. The goal of this game is to get as many points by clicking the animal. The video shows key features of my program, such as buttons to navigate throughout different screens, score increase, lives reduction. In the beginning, 3 lives are given but the player loses a life when they click elsewhere other than the animal. The player can buy items in the store using the coins they have earned. The player is rewarded with a butterfly when they reach above 1000 coins. The program restarts when the player loses all of their lives.

2b)

This program 'FrogFly' game is an extension of a clicker-game that I built in class and developed solely by me. First I started by creating multiple screens and designing the backgrounds, font styles, buttons. Then I added onEvent handlers to the buttons to navigate through the app. After, I added a transparent background image of frog, and hearts to show the lives remaining to the game_screen. One major difficulty I encountered while developing this program is how to change the object frog to a butterfly after the player reached a certain score. I figured it out by declaring a global variable score and lives to keep track. One opportunity I recognized during the development is to add a pause button where the player can resume the game, access the store, or quit the game.

2c)



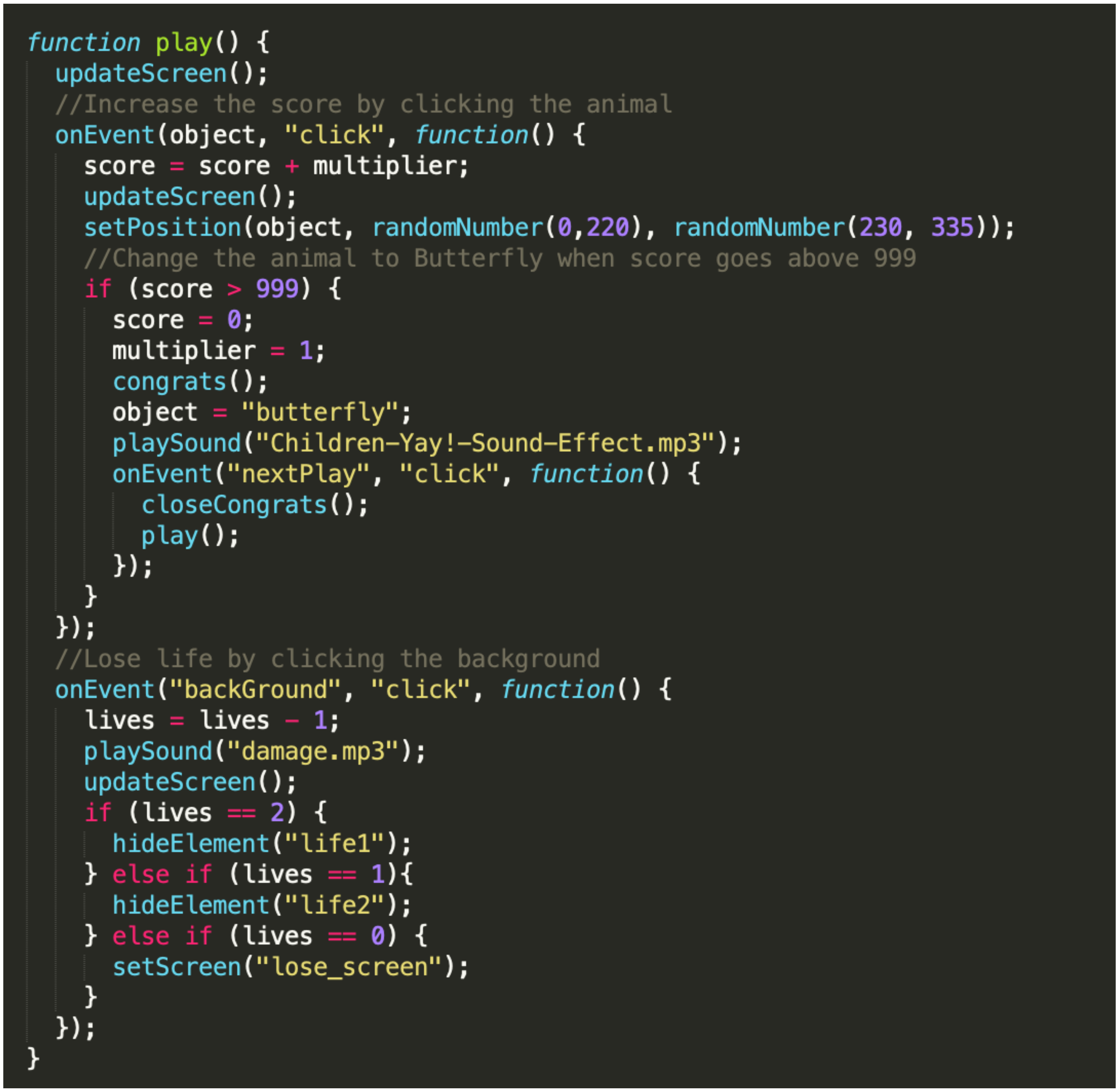
```
function play() {
  updateScreen();
  //Increase the score by clicking the animal
  onEvent(object, "click", function() {
    score = score + multiplier;
    updateScreen();
    setPosition(object, randomNumber(0,220), randomNumber(230, 335));
    //Change the animal to Butterfly when score goes above 999
    if (score > 999) {
      score = 0;
      multiplier = 1;
      congrats();
      object = "butterfly";
      playSound("Children-Yay!-Sound-Effect.mp3");
      onEvent("nextPlay", "click", function() {
        closeCongrats();
        play();
      });
    }
  });
  //Lose life by clicking the background
  onEvent("backGround", "click", function() {
    lives = lives - 1;
    playSound("damage.mp3");
    updateScreen();
    if (lives == 2) {
      hideElement("life1");
    } else if (lives == 1){
      hideElement("life2");
    } else if (lives == 0) {
      setScreen("lose_screen");
    }
  });
}
```

```
function resetGame(){
  setScreen("welcome_screen");
  startingValues();
  updateScreen();
  resume();
}

function startingValues(){
  score = 0;
  lives = 3;
  doubleCost = 500;
  showElement("life1");
  showElement("life2");
  showElement("life3");
  multiplier = 1;
  insectsCount = 0;
  lifeCount = 0;
  doubleCount = 0;
  insectsCost = 20;
  lifeCost = 100;
}
```

```
function updateScreen(){
  setProperty("score_lbl", "text",("💰"+score));
  setText("multipliere","x" + multiplier);
  setProperty("number_Points", "text",("💰"+score));
  setProperty("multiplier", "text",("x" + multiplier));

}
```

My main algorithm is inside the function play(), which is developed independently by me. I put onEvent handlers in this function to implement the sequence of code when a button/image is clicked. This allows the player to actually play the game, clicking the animal and increase the score by addind with the 'multiplier,' which they buy from the store using the coins . Additionally, this function makes the player lose a life when they click elsewhere. This algorithm achieves the intended purpose of my overall program, which is score increase and lives reduction. This algorithm has some sub-algorithms to make the program efficient. These include resetGame(), startingValues(), updateScreen(). Each of them has an individual purpose for the program. Function 'resetGame()' makes the program reset itself, function 'startingValues()' sets all values to their original when the program restarts, function 'updateScreen()' is crucial because it shows the values on the user interface. Combing all of the sub-algorithms made my program simpler and quicker.

2d)



```
function play() {
  updateScreen();
  //Increase the score by clicking the animal
  onEvent(object, "click", function() {
    score = score + multiplier;
    updateScreen();
    setPosition(object, randomNumber(0,220), randomNumber(230, 335));
    //Change the animal to Butterfly when score goes above 999
    if (score > 999) {
      score = 0;
      multiplier = 1;
      congrats();
      object = "butterfly";
      playSound("Children-Yay!-Sound-Effect.mp3");
      onEvent("nextPlay", "click", function() {
        closeCongrats();
        play();
      });
    }
  });
  //Lose life by clicking the background
  onEvent("backGround", "click", function() {
    lives = lives - 1;
    playSound("damage.mp3");
    updateScreen();
    if (lives == 2) {
      hideElement("life1");
    } else if (lives == 1){
      hideElement("life2");
    } else if (lives == 0) {
      setScreen("lose_screen");
    }
  });
}
```

My abstraction includes in the function play(), which is the main part of the program that I wrote by myself. This function allows the player to increase their score, change the animal when the score reaches above 999, and lose a life when the player clicks elsewhere. Also, this code moves the current animal to a random location on the canvas. This allowed me to reduce the complexity by placing all the crucial parts of the program inside one function.