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 is written in JavaScript using the App Lab programming environment. The purpose of my program is to provide entertainment to the user through playing a game. You click the arrow keys to move the plane icon, when it overlaps with the garbage can you win the game. However if you do not complete the objective within 15 seconds you run out of time, which is what occurs the first time it is played in video. The goal is to get the highest score by completing the game as fast as possible.

2b)

I independently developed this program. While making my game I encountered some problems and identified solutions. First I had to make the arrow buttons allowing my plane icon to move. Initially the goUp variable got its value outside of the function, however the plane would only move once. I worked with a peer to collaboratively find a solution. I assigned the goUp variable its value within the function. Every time the OnEvent was initiated it would get the current placement of the plane so the button could work repeatedly. Then I designed the screens. I made the timer end the game after 60 seconds. After peer feedback I made the timer run for 15 seconds to increase the difficulty and added a score for incentive. Then I made a function to check if the plane was above the garbage can by checking the coordinates. Later I had trouble resetting the timer when replaying the game. First I simply set the variable seconds to 0, but when I ran the program it didn't work. Independently I found a solution when I the created the timer and realized I used a timedLoop. I tried using stoptimedLoop, and the timer reseted properly.

2c)

```
82      }, 2000);
83  }
84      //Times Up
85      //if seconds is 15 write Times Up and change to time up screen
86      //reset text to nothing
87  if (seconds == 15) {
88      stopTimedLoop();
89      console.log("It has been 15 seconds");
90      setScreen("loseScrn");
91      // increase variable counting the number of times you have played the game by 1
92      gamePlay = gamePlay + 1;
93      if (score == 0) {
94        // increases variable counting number of times user has lost by 1
95        // plays sound 'Game Over'
96        losePlay = losePlay + 1;
97        playSound("sound://category_male_voiceover/game_over_male.mp3", false);
98      }
99      if (losePlay == 4) {
100       //doubles players losePlay score if they have lost 4 times
101       losePlay = losePlay * 2;
102       setScreen("lost4Scrn");
103     }
104 }
105   });
106 });
```

My main algorithm I selected combines checking if the time is 15 and checking if the score is zero to carry out the code necessary if you lose the game.

The main algorithm exists on lines 87 to 104 and works by checking if seconds is 15, then it changes the appropriate screen and sets the text. It also stops the times loop so the timer will stop counting. It also adds 1 to the gamePlay variable, which keeps track of how many times the game is played. If you have run out of time you have still completed the game.

The first sub algorithm is my nested if statement that exists on lines 93 to 98. It asks whether or not the score is 0. If the score is 0 the code will add 1 to the variable losePlay, which counts the number of times the user has lost the game. It will also play a sound saying 'Game Over'.

The second sub algorithm is the second nested if statement and exists on lines 99 to 103. When you lose the game, it checks if your losePlay amount if equal to 4. If your lost 4 times it changes the screen to lose4Scrn and doubles the players losePlay score.

2d)

```
  });
});

//CheckPosition Function
function checkPosition(){
  X = getXPosition("plane");
  Y = getYPosition("plane");
  //There are two postions that the plane can be for the code
  //to consider the plane to be 'in the garbage can
  //Gets the coordinates(vars) and checks ==
  if ((X == 35 && Y == 30) || (X == 5 && Y == 30)) {
  //if you're in the right place go to win screen, stop counting seconds
  //increases variable counting the number of times you have won by 1
  winPlay = winPlay + 1;
  console.log("WIN");
  setScreen("winScrn");
  stopTimedLoop();
  //write your final score
  setText("FinalScreBx", "With a Score of: " + score);
  //if you have completed it under 10 seconds it says you are fast
  //otherwise you're slow
  if (seconds > 10) {
    hideElement("fastTxt");
    showElement("slowTxt");
  }
}
}

//goHome function
//reset game items (planes, seconds, score, screen
```

The name of my abstraction is checkPosition. It uses an if statement to check if the coordinates of the plane are the same as the garbage can. If they are it takes you to the winning screen, if not it continues the game. I implemented this abstraction into all the arrow keys. Without making the function I would have had to write the entire code for it 4 times because I have 4 arrows. Without the abstraction 60 extra lines of code would be added, and considering the complexity of the code it would be very easy to make a mistake in the code. This area of my program is also very valuable and essential in allowing my code to work. Without checking the coordinates you could never win the game and the time would run out every time. If a mistake would made it would severely damage the functionality of my game. The extra code also would make it hard to understand the other aspects of the code because it would be very easy to get lost within the entire program. The abstraction helps to manage complexity and if I need to the modify the code I only need to change it once within the function.