Create PT - Written Response Template

Assessment Overview and Performance Task Directions for Students

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)

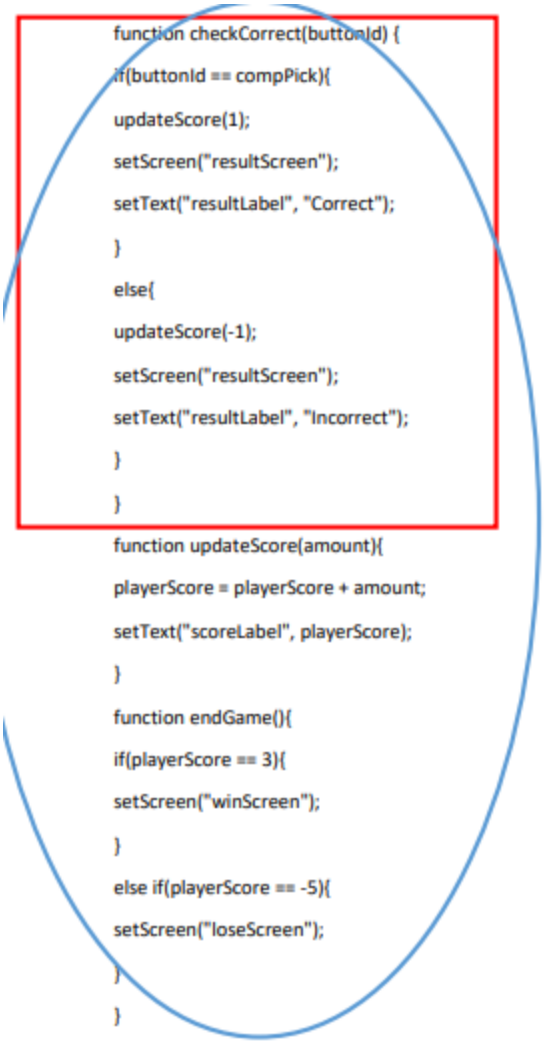
The programming language used to write the code is JavaScript. The purpose of my program is to guess which option the computer is going to randomly input. The computer stores a random number 1-3. The user, inputs a number 1-3 by clicking on one of the fruits. The values are compared and you either gain or lose points based on the result. My video illustrates the "game." The video shows me clicking on a fruit. Whether I got it right or wrong determines whether my score goes up or down. In the end I lost because I got -5 points before I could get 3 points.

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)*

Both developments were individual. When writing my code I ran into the trouble of what to name the pictures so that they can be randomly picked by the computer. At first I named the pictures fruit names because that is what they were displaying. This made it difficult for me to randomize the computer pick. I tried using an array and then randomizing it with the randomNumber command by using the array length. I realized that I could increase efficiency by naming the pictures as numbers. I assigned each picture a number and then used randomNumber so the computer could have a randomly assigned picture. Then the computer number could be compared to a user input of 1, 2, or 3. Another difficulty I had was with the development of the user interface. On the result screen the score would not display. At first I thought it was a problem with my code. I made sure the link from the functions was correct and that the number should display. Afterwards I realized that the text box was not large enough for the number to display.

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



```
function checkCorrect(buttonId) {
if(buttonId == compPick){
updateScore(1);
setScreen("resultScreen");
setText("resultLabel", "Correct");
}
else{
updateScore(-1);
setScreen("resultScreen");
setText("resultLabel", "Incorrect");
}
}
function updateScore(amount){
playerScore = playerScore + amount;
setText("scoreLabel", playerScore);
}
function endGame(){
if(playerScore == 3){
setScreen("winScreen");
}
else if(playerScore == -5){
setScreen("loseScreen");
}
}
```

Written Response

The “checkCorrect” algorithm checks to see if the user input matches the random computer number. The screen is switched to the result screen. If the inputs match, the score is updated to add 1 and the “resultLabel” displays “correct.” If the inputs don’t match, the score is updated to subtract 1 and the “resultLabel” displays “incorrect.” The “updateScore” algorithm sets “playerScore” equal to “playerScore” plus an “amount.” The “scoreLabel” is set to display “playerScore.” The “endGame” algorithm checks to see if “playerScore” is equal to 3. If “playerScore” is equal to 3, the screen is set to the “winScreen.” If “playerScore” is not equal to 3 the algorithm checks if “playerScore” is equal to -5. If it is equal to -5, the screen is set to “loseScreen.” If “playerScore” is not equal to 3 or -5, the algorithm does not run. The “checkCorrect” algorithm works in combination with the “updateScore” algorithm by setting the parameter for “updateScore” to run. Once the parameter is set, “playerScore” can be updated and the result can be displayed. These algorithms work in combination with the “endGame” algorithm by updating the “playerScore” every time so that it can be checked whether the game has ended.

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

```
function checkCorrect(buttonId) {
if(buttonId == compPick){
updateScore(1);
setScreen("resultScreen");
setText("resultLabel", "Correct");
}
else{
updateScore(-1);
setScreen("resultScreen");
setText("resultLabel", "Incorrect");
}
}
```

Written Response

The abstraction helped to manage the complexity of my program by storing a set of code that I would use multiple times in the future. By storing it once and then calling it in all the places I need it in the code, I manage the complexity of the code. I take the complex code and make it simpler by placing it in a function. The function is complex while calling the function is simple.

Export or save this document as a PDF and turn in to the AP Digital Portfolio along with your **Video** and **Program Code** (separate files).