Create — Applications from Ideas
Written Response Submission Template

Submission Requirements

2. Written Responses

Submit one PDF document in which you respond directly to each prompt. Clearly label your responses **2a – 2d in order. Your response to all prompts combined must not exceed 750 words, exclusive of the Program Code.**

Program Purpose and Development

2a. Identify the programming language and identify the purpose of your program. Explain your video using one of the following:

- A written summary
- of what the video illustrates OR
- An audio narration in your video. If you choose this option, your response to the written summary should read, “The explanation is located in the video.”

(Approximately 150 words)

Insert response for 2a in the text box below.

My program is an interactive game created with Javascript. The purpose of this game is to create something fun that interacts with the user by incorporating an adventure that the player completes by performing certain tasks or winning certain games within each adventure. Once all of the subgames on each level of the app are won, the user completes the adventure. The video shows one of two adventures that are incorporated in my game to display how the program tracks score within each mini game and on each overall level/adventure to determine when the overall game/adventure is won. The video also shows the functionality of the games and how they are played, in addition to the storyline format that the adventures follow.

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; the second could refer to either collaborative or independent program development. *(Approximately 200 words)*

Insert response for 2b in the text box below.

I wrote this program, first by creating an identity for the user by incorporating user choice and a user input box to allow for personalization, and then by creating worlds with mini games within them for the user's chosen character to complete. My first problem arose when I tried to carry the username input from the character choice screen throughout the rest of the game. Because I had initially used a local variable, the user input was only recorded to the screen on which it was provided, instead of to the entire program. To resolve this issue, I changed the username input to a global variable that was able to be called on multiple screens in the program. I encountered a similar issue when I attempted to move the user's chosen character between screens. To make it appear as though the image traveled to the new screen as the user switched screens, I created a variable that stored the value of whichever character the user clicked on the first screen (i.e. boy= 1 when the boy image was clicked) and then used an algorithm that incorporated an if statement to set other images throughout the program to the user's character of choice. Both of these difficulties were handled independently.

2c. Capture and paste an image or images of your program code segment that implements the most complex algorithm you wrote. (marked with a **color border** below)

1.	A.
	
	onEvent("coin", "click", function(event) {
	treasureScore = treasureScore+1;
	setPosition("coin", randomNumber(105, 220), randomNumber(190, 250));
	setText("coinCollected", "Number of coins collected: "+treasureScore);
	if (treasureScore==20) {
	setScreen("treasureWin");
	gotCoins = 1;
	}
	});
	onEvent("treasureContinue", "click", function(event) {
	treasureScore = 0;
	setScreen("underwaterAdventures");
	didWinUnderwater();
	if (gotCoins==1) {
	setText("coins", "Coins? Yes");
	}
	});
		B.
		
		onEvent("shell", "click", function(event) {
		setScreen("fishWin");
		scale = 1;
		});
		onEvent("fishContinue", "click", function(event) {
		setScreen("underwaterAdventures");
		didWinUnderwater();
		if (scale==1) {
		setText("scales", "Scale? Yes");
		}
		});

2.	function didWinUnderwater() {
	if (gotCoins==1&&scale==1) {
	setPosition("fishTalk", 25, 150, 210, 130);
	showElement("fishTalk");
	setText("fishTalk", "You have collected all of the elements you need to deliver to the shark. Would you like to proceed?");
	showElement("yesUA");
	showElement("noUA");
	setPosition("underwaterCharacter", 250, 205);
	reset(gotCoins, scale);
	}
	}	

Your algorithm should integrate several mathematical and logical concepts.
Describe the mathematical and logical concepts used to develop the algorithm.
Explain the complexity of the algorithm and how it functions in the program.
(Approximately 200 words)

Insert text response for 2c in the plain box below.

In the adventure “Underwater Adventures” there is a mini game in which the user must collect 20 coins to win. Once this task is complete, the user must play a second mini game in which he/she collects a scale. When both mini games are won, the Underwater Adventures level is complete. The code to execute this concept incorporates a variety of variables and functions to create an algorithm that calculates when the treasure game and scale games have been won, and when the overall level has been completed. The first set of code shown above (1.A.) is the function that runs the treasure game. Each time the coin is clicked, the variable *treasureScore* increases by 1. Once the score reaches 20, another variable, *gotCoins*, increases by 1 and the screen switches to a win screen with a continue button. The scale game works similarly, with a variable *scale* increasing by one if the game is won (1.B.). If the user clicks the continue button after winning either game, the second piece of code shown (2.) uses an algorithm to check if both games have been won by calling the function didWinUnderwater, which uses an if statement to determine if both gotCoins and scale have the value of 1.

2d. Capture and paste an image or images of the program code segment that contains an abstraction you developed (marked with a matching **blue color border** below)

1.
```
function setUpBlastOff() {
 mars = 0;
 jupiter = 0;
 hideText("alienTalk","okMars");
 hideText("alienTalk", "okJupiter");
 setText("saveMars", "Saved Mars? No");
 setText("saveJupiter", "Saved Jupiter? No");
 setPosition("spaceCharacter", 90, 285);
 setCharacter("spaceCharacter");
 showText("alienTalk", "playBlastOff");
 showElement("blastOffInstructions");
 setText("alienTalk", ("Hello "+username)+"! All of the planets in space are becoming uninhabitable! I guess I'll
just have to invade Earth.");
 setText("blastOffInstructions", "Prevent the alien from invading Earth by making the other planets in the
galaxy safe for life. Click on the planets to explore them.");
}
```

2.
```
function hideText(textArea, button) {
 hideElement(textArea);
 hideElement(button);
}
```

3.
```
function showText(textArea, button) {
 showElement(textArea);
 showElement(button);
}
```

4.
```
function setCharacter(image) {
 if (boyClick>=1) {
   setImageURL(image,
"https://www.carstickers.com/prodimages/1033_blue_sky_family_boy_stick_figure_sticker_decal.gif");
 } else {
   setImageURL(image,
"https://www.carstickers.com/prodimages/1032_blue_sky_family_girl_stick_figure_sticker_decal.gif");
 }
}
```

Your abstraction should integrate mathematical and logical concepts.
Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

Insert text response for 2d in the plain box below.

My code uses abstraction by incorporating a variety of functions to condense repeated code into smaller pieces. These functions can then be called within other functions or onEvents to make the code easier to read and understand. The function setUpBlastOff (1.) calls the functions hideText (2.), showText (3.), and setCharacter (4.), which are used to reduce the number of lines of code contained within the function. The setCharacter function (4.) is also an abstraction because it uses an if statement to check the value of the two character variables (boy and girl) and uses this logic to set the on-screen character to the one that the user chose. Overall, the function setUpBlastOff is an abstraction because it is called multiple times in the program to reset the 321BlastOff screen when the level is either won or quit. This prevents the programmer from having to retype the same several lines of code each time they want to reset the screen. These abstractions make my program more manageable because they take repeated sections of code that would add significant complexity to my algorithms and reduce them into single functions. For example, the setUpBlastOff function would require at least 12 lines of code each time it was called if it were not condensed into a function.