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.

This program was created using JavaScript. It's intended to be a turn-based game where players can progress through levels by gaining EXP. The video I've provided displays one of the main and essential features of my program, the attack system. At the start, I showed you my starting health, EXP, and gold. Then, I went into level two and displayed the stat check button and both attack buttons. The celestial attack does more damage than the twilight attack does in this level. Once the enemy was defeated, I showed you the new values for health, EXP, and gold. After that, I bought the simple dagger from the shop and displayed how much more damage it does than a normal attack. We can also see how the program displays the amount of HP you lose when the enemy attacks you after your attack. They also have the probability to miss their attack.

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.

This entire program was undertaken independently. I ran into several complications among the code that I had to figure out on my own. One of the issues that I came across involved configuring how I was going to unlock levels appropriately whenever a player gets a certain amount of EXP. At first, I was going to make a set of code that would keep checking if the amount of EXP reached a required amount or not. I found that it was more efficient just to create a function and run it whenever it was necessary. Now, when a player returns back to the level screens, the program will check to see if the EXP has reached the highest level, and if not, it checks the next highest level and so on. I also had to figure out how I was going to adjust the enemy's health correctly for each level. I decided to make a variable that would be the enemy's health for all levels and just set the value whenever a level was entered. On top of that, a function was made to update the display of the new level and show the appropriate values for everything on screen.

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)

```
onEvent("twiAtk6", "click", function(event) {
  playerAttack(85, 65, 90, 100, "eneHPVal6", "youMiss6");
  if (health <= 0) {
    playerDeath(100);
  } else if (enehealth <= 0) {
    enemyDeath(5000, 1500, 10000);
  } else {
    enemAttack(91, 200, 93, 260, "healthVal6", "hpLose6", "atkMsg6", "missMsg6");
  }
});
```

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.

This is one of the most complex algorithms that I've written for my program. I had to repeat this algorithm at least twelve separate times with different parameters in order to fulfill working attack functions for all levels in the game. The algorithm includes not one, but four functions inside of it with it's own individual parameters that have to be changed for each level. The code starts out by initiating playerAttack which rolls a number from 1 to 100. It's declared either a critical, a basic attack, or a miss. If the attack is critical, it subtracts the critical value rather than the basic value from the enemy's health on top of what kind of damage is added on from any weapons you might have. Then, it checks to see if the enemy's health is at 0 yet and if it is, it will send you to the level screen again and reward you with gold, EXP, and health. If your health is at 0, it will only give you some health. Otherwise, the function enemyAttack is run and the enemy rolls a number and attacks you instead, also with a chance of a critical and a miss.

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)

```
function levelUnlock() {
if (exp >= 10000) {
  hideElement("sixLk");
  showElement("sixOpen");
} else if (exp >= 8000) {
  hideElement("fiveLk");
  showElement("fiveOpen");
} else if (exp >= 5000) {
  hideElement("fourLk");
  showElement("fourOpen");
} else if (exp >= 1500) {
  hideElement("threeLk");
  showElement("threeOpen");
} else if (exp >= 500) {
  hideElement("twoLk")
  showElement("twoOpen")
}
}
```

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.

An abstraction is something that can be used to compress code down and simplify it. This means we do not have to repeat a long block of code over and over again and repeat it in the program. Instead, we can create an abstraction and simplify the process a lot more. One of the abstractions in my program is the levelUnlock() function. Every time the program returns to the level selection screen after completing a level, it runs this abstraction and goes through the requirements for each level to be unlocked. Now, instead of having to individually go through and check the EXP requirements for every level whenever a level is completed, I can easily use the abstraction levelUnlock() to simplify the complexity of my program. Before I came up with this abstraction to use in my program, I was going to have to create a big else if statement for every time the enemy was attacked and their health reached zero which would send them back to the selection screen. The else if statement would have been redundant to keep repeating for every attack button in the program, so crafting an abstraction made it a lot easier to manage.