**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.**

**Program Purpose and Development**
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)

---

**Student Response**

![Video Thumbnail](https://youtu.be/zT1XNAS4oGE)

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

---

**Scoring Guidelines**

<table>
<thead>
<tr>
<th>Row and Task</th>
<th>Decision Rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 1 Response 2A</td>
<td>Response earns the point if it explains the function of the program instead of identifying the purpose.</td>
</tr>
<tr>
<td>Row 2 Decision 2B</td>
<td>Response earns the point if the illustrated feature runs, even if it does not function as intended.</td>
</tr>
<tr>
<td>Row 3 Decision 2C</td>
<td>Response earns the point if the video includes a narration or some form of closed captioning that addresses the purpose of the program.</td>
</tr>
</tbody>
</table>

**Do NOT award a point if any one of the following is true:**
- a video is not submitted;
- the video does not illustrate the feature mentioned in the response; or
- the video does not illustrate the running of the feature.
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)*

<table>
<thead>
<tr>
<th>Student Response</th>
<th>Scoring Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row and Task</strong></td>
<td><strong>Decision Rules</strong></td>
</tr>
<tr>
<td><strong>Row 2 - Response 2B</strong></td>
<td>Do NOT award a point if any one of the following is true:</td>
</tr>
<tr>
<td>Describes or outlines steps used in the incremental and iterative development process to create the entire program.</td>
<td></td>
</tr>
<tr>
<td><strong>Row 3 - Response 2B</strong></td>
<td>Response earns the point if it identifies two opportunities, or two difficulties, or one opportunity and one difficulty AND describes how each is resolved or incorporated.</td>
</tr>
<tr>
<td>Specifically identifies at least two program development difficulties or opportunities.</td>
<td>Do NOT award a point if any one of the following is true:</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
<tr>
<td>Describes how the two identified difficulties or opportunities are resolved or incorporated.</td>
<td></td>
</tr>
<tr>
<td>The response earned a point for this row.</td>
<td>The response describes two difficulties are encountered and how both were resolved. The first difficulty</td>
</tr>
</tbody>
</table>

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 created to calculate the probability of the enemy missing their attack. This function is used to adjust the enemy's health for each level.
made to update the display of the new level and show the appropriate values for everything on screen. (200 words)

described is how to unlock levels when a certain amount of EXP is reached. This is resolved by creating a function that checks to see if the EXP has reached the highest level. The second difficulty described is the tracking of the health of the enemy. This is resolved by using a variable and a function to update the variable accordingly.

2c. Capture and paste a program code segment that implements an algorithm (marked with an oval in section 3 below) 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)

```javascript
function playerAttack(num) {
    if (num < 1) {
        return "miss";
    } else if (num < 10) {
        return "basic";
    } else if (num < 50) {
        return "critical";
    } else {
        return "miss";
    }
}

let eneHealth = 1000;

function checkLevel() {
    if (eneHealth <= 0) {
        alert("You have won!");
    } else if (eneHealth < 100) {
        alert("Your health is low."
    } else if (eneHealth < 500) {
        alert("Your health is critically low."
    } else if (eneHealth < 1000) {
        alert("Your health is very low."
    } else {
        alert("You have no more health left."
    }
}
```

Student Response

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

<table>
<thead>
<tr>
<th>Row and Task</th>
<th>Scoring Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Row 4</td>
<td>Do NOT award a point if any one of the following is true:</td>
</tr>
<tr>
<td>Response 2C</td>
<td>● the algorithm consists of a single instruction;</td>
</tr>
<tr>
<td></td>
<td>● the code segment consisting of the algorithm is not included in the written responses section or is not explicitly identified in the program code section; or</td>
</tr>
<tr>
<td></td>
<td>● the algorithm is not explicitly identified (i.e., the entire program is selected as an algorithm, without explicitly identifying the code segment containing the algorithm).</td>
</tr>
<tr>
<td>The response earned a point for this row.</td>
<td></td>
</tr>
<tr>
<td>The selected code segment for twiAtk6 implements an algorithm.</td>
<td></td>
</tr>
</tbody>
</table>

Row 5
Response 2C
The algorithm being described can utilize existing language functionality, or library calls. Response earns the point even if the algorithm was not
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. (199 words)

<table>
<thead>
<tr>
<th>Row 6</th>
<th>Response 2C</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Selected code segment</strong></td>
<td>implements an algorithm that includes at least two or more algorithms.</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td><strong>Explain how the included algorithms functions.</strong></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>Describes what the included algorithms does in relation to the overall purpose of the program.</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td><strong>Explains how the included algorithms functions.</strong></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td><strong>Describes what the included algorithms does in relation to the overall purpose of the program.</strong></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td></td>
</tr>
</tbody>
</table>

**The response earned a point for this row.**
The selected algorithm includes logical concepts through the use of an if statement. The response explains how the algorithm functions. The response states, “The code starts out by initiating playerAttack… 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…”
The response describes what the algorithm does in relation to the overall program. The response states, the algorithm is used “in order to fulfill working attack functions for all levels in the game.”

**Row 6**
**Response 2C**

**Selected code segment**

- implements an algorithm that includes at least two or more algorithms.

**AND**

- At least one of the included algorithms uses mathematical or logical concepts.

**AND**

- Explains how one of the included algorithms functions
The response earned a point for this row.
The selected code segments includes two or more algorithms, specifically playerAttack, playerDeath, enemyDeath, and enemAttack.
The response indicates that playerAttack includes mathematical and logical concepts. It states that the procedure “rolls a number from 1 to 100”, includes conditionals for “if the attack is critical”, and “subtracts the critical value… from the enemy’s health”.
The response explains how playerAttack functions independently. The response states, playerAttack “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.”

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 below). This abstraction must integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program. (Must not exceed 200 words)

```javascript
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")
    }
}
```

**Student Response**

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

<table>
<thead>
<tr>
<th>Scoring Guidelines</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Row and Task</strong></td>
</tr>
<tr>
<td>Row 7</td>
</tr>
</tbody>
</table>
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. (200 words)

<table>
<thead>
<tr>
<th>Response 2D</th>
<th>as creating a list to represent a collection (e.g., a classroom, an inventory), would earn this point.</th>
</tr>
</thead>
</table>
| Selected code segment is a student-developed abstraction. | **Do NOT award a point if any one of the following is true:**  
  ● the response is an existing abstraction such as variables, existing control structures, event handlers, APIs;  
  ● the code segment consisting of the abstraction is not included in the written responses section or is not explicitly identified in the program code section; or  
  ● the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction). |

The response earned a point for this row.  
The selected code segment is a student-developed function levelUnlock.

---

**Row 8 Response 2D**

Explains how the selected abstraction manages the complexity of the program.

Responses should not be penalized for explanations of abstractions that are not developed by the student.

**Do NOT award a point if any one of the following is true:**  
● the explanation does not apply to the selected abstraction; or  
● the abstraction is not explicitly identified (i.e., the entire program is selected as an abstraction, without explicitly identifying the code segment containing the abstraction).

The response earned a point for this row.  
The response explains how the abstraction manages complexity. The response states, “Instead of having to individually go through and check the EXP requirements for every level whenever the level is completed, I can easily use the abstraction levelUnlock() to simplify the complexity of my program.”