2a. The programming language I used is JavaScript. The purpose of my program is for users to have fun answering challenging riddles. By answering questions right, the user earns points that go towards their final score. If they have enough points by the end, they win the game. The video shows an example of a user correctly answering a riddle and incorrectly answering a riddle, along with the increase/deduction in points.

2b. First, I came up with the idea of having users answer a series of questions and gaining/losing points for right/wrong answers. From there, I knew I needed two global arrays, one that contained questions and one that contained answers. I also needed two global variables to set up the points and current index.

I split the game into four functions: a gameStart that reset variables and set up proper screens, a gamePlay which switched the question through adjusting array position, a checkCorrect that checks if the user answer is equal to the answer array at the current index, and an endGame which checks the index to see if any questions are left.

I ran into a problem with the current index variable. The game kept skipping my first question when I had the index set to 0 because my gamePlay function added one to it at the start. Changing it to -1 fixed my problem. I also had a problem with my check button. I first had my
event function nested in my `checkCorrect` function, but I realized the code was stuck in an endless loop. Separating the two kept the code from repeating itself and skipping questions.

2c.

This algorithm is essential to determining when the game should end and preventing out of bound errors.

The `checkEnd()` function checks the current index+1 to the length of the array, as if both are equal no questions are left and the game should end. If the current index+1 and length are equal, the function then determines if the user should win or lose. If the user's points are equal to the length of the array, meaning they got every question right, the function switches the screen to the win screen and resets the variables for the next game. If the user's points are less than the
length, meaning they missed something, the function switches the screen to the lose screen and resets the variables for the next game.

If the current index+1 and length aren't equal, meaning the user hasn't answered all the questions, the function calls the gamePlay() function and adjusts the current index to the next question. The user inputs their result, presses the check button which calls for their answer to be checked against the real answer. If correct, it awards points. If incorrect, it deducts. Then the program checks for the end, and the cycle continues.

2.d

```javascript
function checkCorrect() {
    var UserAnswer = getText("UserAnswer");
    var Answer = Answers[currentIndex];
    console.log(UserAnswer + " " + Answer);
    // checks to see if the user answer matches the answer array key
    if (UserAnswer==Answer) {
        console.log("correct");
        Points+=1;
        setText("points", "Points: " + Points);
        console.log("Current Answer: " + Answers[currentIndex]);
    }
    else {
        console.log("incorrect");
        Points-=1;
        setText("points", "Points: " + Points);
        console.log("Current Answer: " + Answers[currentIndex]);
    }
}
```
An abstraction I created was a function to determine if a question was correct. Two variables, User Answer which grabs text from the user answer textbox and Answer which grabs text from the answers array, are checked against one another. If the user’s answer matches the correct answer, the game adds a point and sets the new point value. If they aren’t equal, the game subtracts a point and sets the new point value. This algorithm makes the code less complex by preventing having to write the same lines of “checking” code after every question in the array.