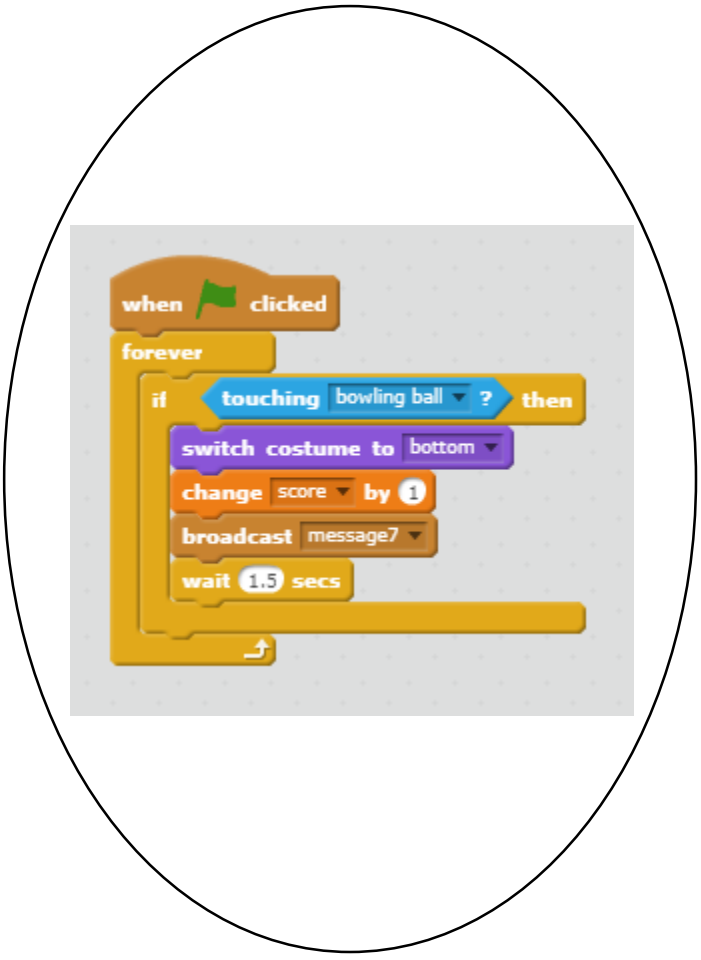
2a. My program is an entertaining bowling game made with Scratch. The purpose of the program is so that the user can play a fun game of bowling by knocking all five pins down. In my video, I demonstrate how the game can be played. In the beginning, there is a host that introduces the game and prompts the user to utilize a number of keys so that they can navigate the bowling ball through the game. The bowling ball must knock all five pins down in order for the game to be over. The space key, up key, down key, right key, or left key can be used to move the bowling ball. Once all the pins are down, the host says "Great job!" and the game is over.

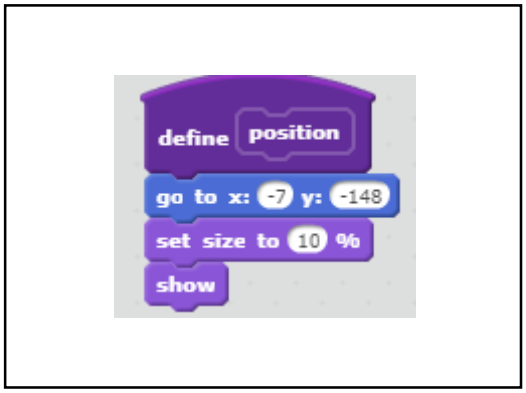
2b. The bulk of my project was created independently, but I had a minimal amount of help from a partner. I started off by planning out how to make the bowling pins get knocked out as the bowling ball went forward to hit them. While doing this, I encountered a huge problem in my code. Every time the bowling ball touched the bowling pins, they didn't disappear although the code specifically stated that they should disappear once they were in contact with one another. My friend helped me with this by suggesting that I should make a code for the bowling pins so that when the bowling pins broadcasted messages and the bowling ball received them, the program would run smoothly and the bowling pins would disappear every time the bowling ball received their individual broadcast message. Another issue I faced was when all the pins were down, the host wouldn't say "Great job!" I resolved this problem by broadcasting a message when the last pin fell down so that when the host received the message, he would say "Great job!"

2c.



This algorithm was one that was fundamentally important in aiding the program to successfully run. In this algorithm, the bowling pin is sensing whether the bowling ball is touching it or not and if it is, the costume of the bowling ball switches from a regular bowling pin to the bottom of a bowling pin to make it look as if the bowling pin has fallen. Next, the score is changed by 1, meaning that every time the bowling ball touches a bowling pin, the score is increased by one point. After that, "message7" is broadcasted so that when the bowling ball receives "message7", it will immediately disappear, wait 1.5 seconds, and then return to its original position so that the user can continue playing the game. This algorithm functions independently because it is solely for the purpose of making sure that the bowling pin is knocked down. Also, in combination with others, this algorithm helps determine the course of the game because if the bowling pin isn't being touched, the algorithm won't run. This algorithm allows other algorithms to be notified that if the bowling pin is being touched, the rest of the program should continue moving forward.

2d.



While I was writing the program for my bowling game, I realized something. Every time the bowling ball touched a bowling pin, I programmed it to hide, go back to its original position, set the size to the original 10%, and then be visible again, I noticed that this code could be made shorter and simpler. I figured that if I made a custom block called "position", I could fit all of those blocks of code into "position" and make it an abstraction block. This helped me significantly because it ensured that every time I would look at the code, I wouldn't get confused to see the same things repeating over and over again and it would make the code seem simpler and more human friendly. It also helped manage the complexity of my program because every time I looked at the block of code "position", it made me understand that the block was simply asking for the bowling ball to be returned back to its original position so that the user could roll the ball again.