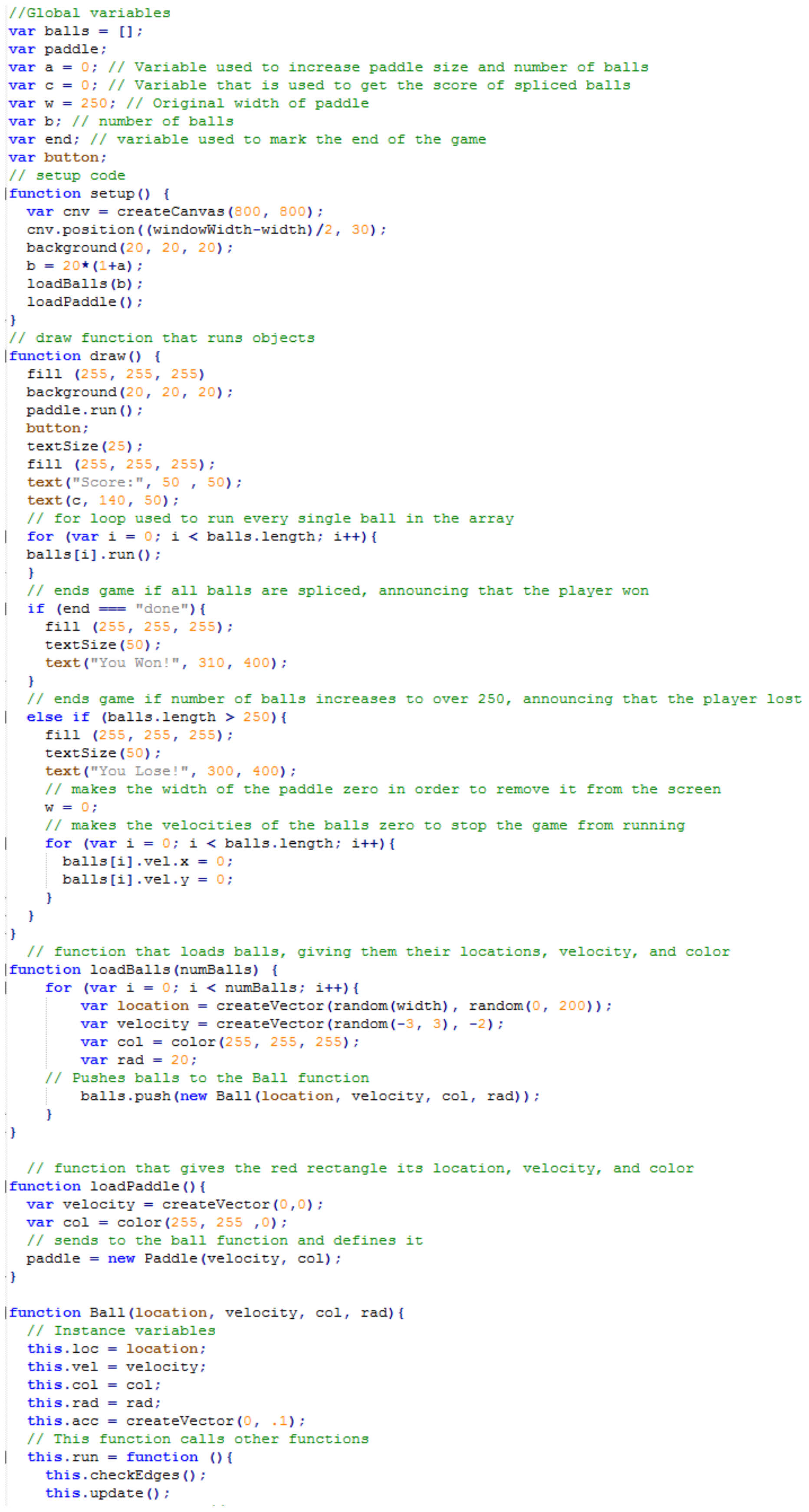
Paddle Game Questions

1. The program is written in javascript. The purpose of the program is to make a game where a paddle moves in the x-direction of the mouse and removes balls when they touch the top of the paddle. If the balls touch the bottom of the paddle, more balls will be added. If the number of balls reaches over 250, the player loses. If the player is able to remove all the balls, they win. The video demonstrates in the first half a winning game and then demonstrates a losing game.
2. The entire program was written independently. I at first was unsure where to begin creating my paddle ball game. But after remembering how I began previous labs, I figured out where to begin. I successfully created a paddle function that moved in the x-direction of the mouse and I made a ball function that created the balls. However, I had a lot of difficulty making the paddle remove the balls when touched at the top of the paddle. Having worked alone during this process, I took a while to figure it out. However, after brainstorming and testing numerous methods, I was able to remove balls when touched at the top of the paddle. Having figured this out, I was easily able to make more balls appear on the screen when they touched the bottom of the paddle and finished the program by creating a conditional statement that determined if the player won or lost.
3. I was able to implement an algorithm that integrates other algorithms and mathematical concepts in order to check for collisions in my paddle ball game and mark the game as finished. The conditional statement integrates variables that find the distances between the paddle and each ball, which are used the find the height the ball from the paddle, all done through mathematical equations. The statement checks if the ball is in between the left and right side of the paddle, checks if its velocity is positive, and checks if the height is less than 10 pixels. If the velocity is positive, meaning the ball is going down, the ball is removed as it is hitting the top of the paddle. Inside the statement, a conditional statement is integrated that checks if all balls have been removed, which will then mark the game as done. Overall, this algorithm checks whether if each ball is 10 pixels near the paddle and if they are moving down, thus splicing them if that is true and checks whether the number of balls is zero, which will mark the game as finished if that is true, notifying the player that they have won the game.
4. My abstraction played a big role in my program. It determined the height of each ball from the paddle which is used to determine whether the ball should be spliced or not. The variables calculated the location of the corners of the paddle, the distance of the corners from the balls, and calculated the height of the ball from the paddle with the pythagorean theorem.

5.



```
//Global variables
var balls = [];
var paddle;
var a = 0; // Variable used to increase paddle size and number of balls
var c = 0; // Variable that is used to get the score of spliced balls
var w = 250; // Original width of paddle
var b; // number of balls
var end; // variable used to mark the end of the game
var button;
// setup code
function setup() {
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(20, 20, 20);
  b = 20*(1+a);
  loadBalls(b);
  loadPaddle();
}
// draw function that runs objects
function draw() {
  fill (255, 255, 255)
  background(20, 20, 20);
  paddle.run();
  button;
  textSize(25);
  fill (255, 255, 255);
  text("Score:", 50 , 50);
  text(c, 140, 50);
  // for loop used to run every single ball in the array
  for (var i = 0; i < balls.length; i++){
  balls[i].run();
  }
  // ends game if all balls are spliced, announcing that the player won
  if (end === "done"){
    fill (255, 255, 255);
    textSize(50);
    text("You Won!", 310, 400);
  }
  // ends game if number of balls increases to over 250, announcing that the player lost
  else if (balls.length > 250){
    fill (255, 255, 255);
    textSize(50);
    text("You Lose!", 300, 400);
    // makes the width of the paddle zero in order to remove it from the screen
    w = 0;
    // makes the velocities of the balls zero to stop the game from running
    for (var i = 0; i < balls.length; i++){
      balls[i].vel.x = 0;
      balls[i].vel.y = 0;
    }
  }
}
  // function that loads balls, giving them their locations, velocity, and color
function loadBalls(numBalls) {
    for (var i = 0; i < numBalls; i++){
        var location = createVector(random(width), random(0, 200));
        var velocity = createVector(random(-3, 3), -2);
        var col = color(255, 255, 255);
        var rad = 20;
    // Pushes balls to the Ball function
        balls.push(new Ball(location, velocity, col, rad));
    }
}

  // function that gives the red rectangle its location, velocity, and color
function loadPaddle(){
  var velocity = createVector(0,0);
  var col = color(255, 255 ,0);
  // sends to the ball function and defines it
  paddle = new Paddle(velocity, col);
}

function Ball(location, velocity, col, rad){
  // Instance variables
  this.loc = location;
  this.vel = velocity;
  this.col = col;
  this.rad = rad;
  this.acc = createVector(0, .1);
  // This function calls other functions
  this.run = function (){
    this.checkEdges();
    this.update();
    this.collision(); // calls collision detection function
    this.render();
  }
  // This function changes the location of the boids
  // by adding speed to location and velocity
  this.update = function(){
    this.vel.add(this.acc);
    this.loc.add(this.vel);
  }
  // This function checks for collision and removes balls
  this.collision = function (){
    for (var i = 0; i < balls.length; i++){
    var m = w;
    // relocates balls if they go out of canvas
    if (balls[i].loc.y > 820){
      balls[i].loc.y = 0;
      balls[i].vel.y = -2;
      }
    // p1 = coordinates of the top left corner of paddle
    var p1 = createVector(paddle.loc.x-(m/2), paddle.loc.y);
    // x1 gets the x-distance between the ball and p1
    var x1 = balls[i].loc.x-p1.x;
    // dist1 gets the distance between the ball and p1
    var dist1 = balls[i].loc.dist(p1);
    // height = the height of the ball from the top of the paddle
    var height = Math.sqrt((dist1*dist1)-(x1*x1));
    // p2 = coordinates of the bottom left corner of paddle
    var p2 = createVector(p1.x, p1.y+20);
    // dist2 gets the distance between the ball and p2
    var dist2 = balls[i].loc.dist(p2);
    // x2 gets the x-distance between the ball and p2
    var x2 = balls[i].loc.x-p2.x;
    // height2 = the height of the ball from the bottom of paddle
    var height2 = Math.sqrt((dist2*dist2)-(x2*x2));
    /* Splices balls if their x distance is between 0 and the paddle's width,
    ** their velocity is positive, and their height is less than 10 pixels from the top of the paddle
    */
    if (balls[i].vel.y > 0 && x1 > 0 && x1 < m && height < 10){
            balls.splice(i, 1);
            c = c+1;
            if (balls.length === 0){
              end = "done";
            }
        }
    /* Adds more balls if the ball's x distance is between 0 and the paddle's width,
    ** their velocity is negative, and their height is less than 10 pixels from the bottom of the paddle
    */
    else if (balls[i].vel.y < 0 && x2 > 0 && x2 < m && height2 < 10) {
            balls.length = 0;
            a = a+1;
            b = 20*(1+a);
            loadBalls(b);
          }
        }
      }
        //checkEdges() reverses speed when the ball touches an edge
  this.checkEdges = function(){
    if(this.loc.x < 0) this.vel.x = -this.vel.x;
    if(this.loc.x > width) this.vel.x = -this.vel.x;
      if(this.loc.y < 0) this.vel.y = -this.vel.y;
      if(this.loc.y > height) this.vel.y = -this.vel.y
  }
  // render() draws the ball at the new location
  this.render = function(){
    fill(this.col);
      ellipse(this.loc.x, this.loc.y, rad, rad);
    }
}
function Paddle(velocity, col){
  // Instance variables
  this.loc = createVector(mouseX, 550);
  this.vel = velocity;
  this.col = col;
  // This function calls other functions
  this.run = function (){
    this.update();
    this.render();
  }
  // This function changes the location of the boids
  // by adding speed to location and velocity
  this.update = function(){
    var mouseLoc = createVector(mouseX, 550);
    this.loc = p5.Vector.lerp(this.loc, mouseLoc, .09)
  }
  // render() draws the ball at the new location
  this.render = function(){
    fill(this.col);
        rect(this.loc.x-(w/2), this.loc.y, w, 20);
    }
}
```