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.

For my create task I used the high level programming language Javascript which utilizes the power of abstractions to make small amounts of code, perform large tasks. The purpose of my program is to make a fun and challenging new version of the classic “Snake Game.” My video illustrates a normal gameplay type scenario in which the player attempts to eat all corgis in the time limit but fails after he/she touches a barrier or tangles the snakes tail. It takes about 5 minutes of focus to actually win so I didn’t include the end screen which says “congratulations, You Won!” after all food have been eaten.

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.
(Approximately 200 words)

My program is a spin off on the classic java “Snake Game” that we all know. However I changed the premise of the game, making it more of a survival game. The goal is to clear all the corgis off the canvas, by eating them, without touching the edges of the canvas or tangling your tail. If either of these happen, the game is over. I started my process by laying out what I want to do in many lines of pseudocode on paper. Once I knew what I wanted to do I began to organize my logic and used my knowledge of four loops and if statements to adjust the loadFood and foodLoc functions. I loaded the number of food I wanted and changed the square food objects to pictures of corgis that I found on the internet, to give the game a friendly aspect. I added a functional timer which counts down from 100 seconds, and it ends the game if time expires by calling the deadGame function. I watched a daniel Shiffman video to figure out this part of the project. The last thing I added was a slider that changes the frame rate for more advanced players.

2c. Capture and paste the program code segment that implements an algorithm (marked with an oval in section 3 below) that is fundamental for your program to achieve its intended purpose. Your code segment must include an algorithm that integrates other algorithms and integrates 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.

This algorithm that is shown here is in the checkLoc function in the sketch.js file. It is the part of the code that controls the splicing of food off the canvas by checking the loc of the snake and the stationary food object and if they touch food.splice is called. It also adds the digit that was food, or a corgi, and makes it a segment of the snake. This is done in snake.segments.push with the help of a vector. This is the meatiest algorithm in all my code and it uses many mathematical concepts to carry out its basic, yet essential job every time the game is played. After I added the timer to my code I had to add a function that would increase the digital score onscreen every time the player successfully eats a corgi. This was a challenge but with instruction from peers I realized all I had to add to this algorithm was add score++. The lines of code that have been hashed out (//) are parts of my pseudocode that explain what the code is doing.

```
//checking location of the food
function checkLoc(){
for(var i = 0; i < food.length; i++){
    var distX = food[i].loc.x - snake.loc.x;
    var distY = food[i].loc.y - snake.loc.y;
    if(distX == (0) && distY == (0)){
      food.splice(i, 1);
      //removes the food
      //would add in a new food if that was the way I wanted it to
be
      loadFood(0);
      snake.segments.push(createVector(0, 0));
      console.log(snake.segments.length)
      score++;
    }
  }
}
```

2d. Capture and paste the program code segment that contains an abstraction you developed (marked with a rectangle in section 3 below). Your abstraction should integrate mathematical and logical concepts. Explain how your abstraction helped manage the complexity of your program.
(Approximately 200 words)

The abstraction that is captured below is the part of the code that calls the flash screen and and the gameover function to end the game. This is only called when the snake is touching a wall and or the tail is tangled. That was the logic behind it but making it into functional code proved to be quite difficult. I asked a friend for help and she walked me through how she did it in another project. After, I made the gameStart and gameOver functions that are called when certain things happen (player dies, or hits refresh). I added the text that I wanted to appear in the strokeWeight and textSize that I desired to end the game. It took me a while to understand the parameters of the canvas and where to situate the flash screen so that it's in the middle. The function deadGame is the main abstraction here and if you are good enough at the game, it won't ever be called.

```
//Game over function
function deadGame(){
  if(snake.status == "true"){
    snake = 0
    score = 0;
    text("Good try bud, refresh for more!", 400, 400);
    loadSnake();
    gameStart();
    gameover();

  }
```

3. Program Code

Capture and paste your entire program code in this section.

- Mark with an **oval** the segment of program code that implements the algorithm you created for your program that integrates other algorithms and integrates mathematical and /or logical concepts.
- Mark with a **rectangle** the segment of program code that represents an abstraction you developed.
- Include comments or citations for program code that has been written by someone else.

Sketch.js

```
var snake;

var food = [];
var numSeg = 1;
var start = "true"
var font;
var score = 0;
var timeRemaining;

//standarn setup
function setup(){
  textAlign(CENTER, CENTER);
  //new framerate
  frameRate(10);
  var cnv = createCanvas(800, 800);
  cnv.position((windowWidth-width)/2, 30);
  background(200, 244, 66);
  loadSnake();
  loadFood(100);
  img = loadImage("corgi.png");
  fSlider = createSlider(0, 50, 10)
  fSlider.position(780, 5);
  frameRate(frames);
  //load in 100 food and have it become depleted
}
//draw functions
function draw(){
  frames = fSlider.value();
  background(200, 244, 66);
  snake.run();
  //score count
  textSize(50);
  text("score: " + score, 120, 50)
  noStroke();
  Score();
  //calls for food function
  for(var i = 0; i < food.length; i++){
    food[i].run();
    //timer function
    textAlign(700, 100);
      textSize(50);
  }
//fucntions
  checkLoc();
  deadGame();
  gameStart();
  Score();
  snake.timer();
}
//checking location of the food
function checkLoc(){
  for(var i = 0; i < food.length; i++){
    var distX = food[i].loc.x - snake.loc.x;
    var distY = food[i].loc.y - snake.loc.y;
    if(distX == (0) && distY == (0)){
      food.splice(i, 1);
      //removes the food
      //would add in a new food if that was the way I wanted it to be
      loadFood(0);
      snake.segments.push(createVector(0, 0));
      console.log(snake.segments.length)
      score++;
    }
  }
}
//snake function call
function loadSnake(){
  var loc = createVector(200, 200);
  var vel = createVector(0, 0);
  snake = new Snake(loc, vel);
}
//loading food into the canvas
function loadFood(numFood){
  for(var i = 0; i < numFood; i++){
    var min = 1;
    //40 * 20 = 800
    var max = 39;
    var locX = (Math.floor(Math.random() * (max - min + 1) + min)) *
20;
    var locY = (Math.floor(Math.random() * (max - min + 1) + min)) *
20;
    var loc = createVector(locX, locY);
    var f = new Food(loc);
    food.push(f);
  }
}
//controls for the snakes direction
function keyPressed(){
  start = "false"
  if(keyCode === 38){
    snake.vel = createVector(0, -20)
  }
  if(keyCode === 40){
    snake.vel = createVector(0, 20)
  }
  if(keyCode === 39){
    snake.vel = createVector(20, 0)
  }
  if(keyCode === 37){
    snake.vel = createVector(-20, 0)
  }
}
```

```
//game over function
function deadGame(){
  if(snake.status == "true"){
    snake = 0
    score = 0;
    text("Good try bud, refresh for more!", 400, 400);
    loadSnake();
    gameStart();
    gameover();

  }
}
```

```
//pop up page of the beginning of the game
function gameStart(){
  if(start == "true"){
    textFont()
    fill(5, 225, 15);
    rect(225, 300, 350, 200);
    fill(0, 0, 0);
    rect(240, 315, 320, 170)
    fill(150, 200, 70);
    textAlign(CENTER);
    textSize(40);
    text("Corgi Chowdown", 400, 435)
  }
}
//score function with win function as well.
function Score(){
  if (score > 99){
  fill(255, 0, 5);
  textAlign(CENTER);
  text("good job", 400, 400);
  }
  snake.timer();
  text(snake.timeRemaining, 0, 100, 1450);
  noStroke();
  if (snake.timeRemaining === 0){
    deadGame();
  }
}
```

Snake.js

```
function Snake(loc, vel){
```

```
//what the snake needs to know
  this.loc = loc;
  this.vel = vel;
  this.segments = [];
  this.status = "false";
  this.timeRemaining = 100

//other functions of the snake
  this.run = function(){
    this.update();
    this.render();
    this.dead();
    this.timer();
  }
//snakes movement
  this.update = function(){
    for(var i = this.segments.length - 1; i >= 0; i--){
      if(i > 0){
        this.segments[i].x = this.segments[i-1].x;
        this.segments[i].y = this.segments[i-1].y;
      }else{
        this.segments[0].x = this.loc.x;
        this.segments[0].y = this.loc.y;
      }
    }
    this.loc.add(this.vel);
    this.loc.x = constrain(this.loc.x, 0, 800-20)
    this.loc.y = constrain(this.loc.y, 0, 800-20)
  }
//render function of the snake
  this.render = function(){
    for(var i = 0; i < this.segments.length; i++){
      fill(69, 68, 89);
      stroke(121, 139, 19);
      rect(this.segments[i].x, this.segments[i].y, 20, 20)
    }
    fill(195, 206, 224);
    rect(this.loc.x, this.loc.y, 20, 20);
  }

//the snakes death function
  this.dead = function(){
    for(var i = 0; i < this.segments.length; i++){
      var distX = this.loc.x - this.segments[i].x;
      var distY = this.loc.y - this.segments[i].y;
      if((distX == 0) && (distY == 0)){
        this.status = "true";
        console.log(this.status);
      }
    }
  }
  this.timer = function () {

     if (frameCount % 40 === 0 && this.timeRemaining > 0) { // if the
frameCount is divisible by 60, then a second has passed. it will stop
at 0
       this.timeRemaining --;
     }
     if (this.timeRemaining === 0) {
    this.dead();
     }

 }

}
```

Food.js

```
//food function
function Food(loc){
//gives the food a random location
  this.loc = loc;

  this.run = function(){
    this.render();
  }
//renders the food on the screen
  this.render = function(){
    fill(255, 0, 0);
    stroke(255);
    image(img, this.loc.x, this.loc.y, 50, 50);
    stroke(121, 139, 19);
  }

}
```