Create – Applications From Ideas
Written Response Submission

Program Purpose and Development

2a)

I created my program in JavaScript using codehs.com. The purpose of the program is for the user to guide a constantly moving square through 5 randomly generated obstacle courses. The user changes the direction the square is moving using the left and right arrow keys. Each obstacle course is a set of four randomly generated lines with randomly placed gaps in the middle. The goal is to make it passed 5 courses. The user loses if they let the square crash into the red walls placed on either side of the canvas. The video first illustrates the fact that if you crash into either of the red walls, you lose, and then it shows that if u successfully guide the square through 5 random obstacle courses, you win.

2b)

I wrote my whole program independently. To help better understand what I had to do I used a process known as top-down design, which is a process where you take a larger problem and break it up into smaller problems and solve each smaller problem individually until you eventually solve the larger problem. I knew that I wanted my program to have obstacles, gravity, movement, etc. So i made each of those aspects separate functions that work together to make the program work. During my development process I was stumped on how to implement gravity into my game. I decided to start off by simply having a function to continuously move the square downward. After I had that working, I expanded the gravity function so it would always check the points at the bottom left and bottom right of the square, and if there was nothing at those points it would move downward, and if there was something at either of those points, it would not move. I saw an opportunity to include a game over screen when the game ends. The game ends in two ways, hitting the red wall and losing, or making to the end and winning. To differentiate one ending from another, I made the end of the game a function that took in a boolean parameter. If the parameter was true, the winning screen would show. If the parameter was false, the losing screen would show.

2c)

```
}

function move(){
    keyDownMethod(direction);
    var elemLeft = getElementAt(player.getX() - 1, player.getY() + PLAYER_SIZE/2);
    var elemRight = getElementAt(player.getX() + PLAYER_SIZE + 1, player.getY() + PLAYER_SIZE/2);
    var elemBottom = getElementAt(player.getX() + PLAYER_SIZE/2, player.getY() + PLAYER_SIZE);
    if(elemLeft == null && elemRight == null){
        player.move(x, y);
    }else{
        if(elemLeft == leftWall || elemRight == rightWall){
            gameOver(false);
        }
    }
    gravity();
    if(player.getX() > getWidth()){
        player.setPosition(getWidth() - PLAYER_SIZE - 1, player.getY());
        x = 0;
    }
    if(player.getX() < 0){
        player.setPosition(0 + PLAYER_SIZE + 1, player.getY());
        x = 0;
    }
    if(player.getY() + PLAYER_SIZE >= getHeight()){
        player.setPosition(player.getX(), 0);
        for(var i = 1; i <= 4; i++){
            var elemLeft = getElementAt(0, i*100);
            remove(elemLeft);
            var elemRight = getElementAt(getWidth(), i*100);
            remove(elemRight);
        }
        obs();
    }
}
```

One algorithm in my program is a function called move which covers all aspects of the game related to movement. Within the move function is another algorithm which is a if-else statement which, in order to allow horizontal movement, checks two points on the canvas, the points on the left and right of the player-controlled square, and if the elements at both of those points equal null, then it is okay to move. Else, if those elements are the red walls, it is a game over. This ensures the square will detect if anything is in the way and thus whether or not to move the square. This uses a getElementAt function provided by codehs.com. Another algorithm in the move function is a for loop that runs every time the player hits the bottom of the screen. Since a new obstacle must be generated every time the player hits the bottom of the screen, this for loop ensures that the previous obstacles are removed so that they don't overlap with the new obstacles. It does this by looping through and getting each element at specified points and removing them. All of my algorithms were written independently.

2d)

```
71  }
72
73  function obs(){
74      for(var i = 1; i <= 4; i++){
75          var leftLength = Randomizer.nextInt(0, getWidth() - PLAYER_SIZE*2);
76          var leftWall = new Rectangle(leftLength, 5);
77          var rightLength = getWidth() - (leftLength + PLAYER_SIZE*2);
78          var rightWall = new Rectangle(rightLength, 5);
79          rect(leftLength, 5, 0, i*100);
80          rect(rightLength, 5, leftLength + PLAYER_SIZE*2, i*100);
81      }
82      runs++;
83      if(runs == maxRuns){
84          gameOver(true);
85      }
86  }
87
```

An abstraction that I developed is a function called obs() which is short for obstacle. This is the function that generates the random obstacle course. each level on the obstacle course consists of two lines of random length. This uses a randomizer class provided by codehs.com. First the length of the left line is generated, and then it uses the length of the left line to calculate the length of the right wall. The length and the location of the lines always ensures a gap of two player lengths between. This function manages the complexity of my program because having the ability to generate a random obstacle course with a single command made it easier to program a continuous cycle of moving through a course, getting to the bottom, and moving to the next course. Furthermore, This function also double as a function that keeps track of the number of times the player has made it through the obstacle course. It plays the winning screen once the function runs 5 times.