First, create a simple HTML page (2.1)

Add the fact we need JavaScript (3.1)

Put the picnic tables in the proper location. You will need to do this by setting the style=attribute on the tables. (2.13, 2.26)

Put a paragraph on the top right of the screen saying “Ants killed: 0.” Give it an ID of scoreboard (2.2, 3.6)

Now, create a bunch of image and hidden form fields enough for two ants. We want enough for two ants, because we want to test that some things happening to one ant (e.g., the ant dies) doesn’t happen to the other ant. You need an image field for both ants, an x xoordinate, a ycoordinate, and whether the ant is alive. These should be defined as array variables. The Y coordinate for both ants should be 0, because all ants start at the top of the screen (2.13, 3.37, 4.9)

Create a hidden form field to indicate whether the game is over. By default, this should be false. Create a hidden form field stating the number of ants. This should be 2 for testing purposes. Create a hidden form field stating the number of dead ants. (3.37)

Create the wiggle animation for each ant. All this needs to do is rotate among the 4 ant pictures from 0 to 3 and then back to 0. The animation should end if the ant is dead or it is game over(7.1, 7.2)

Create a function linked to the HTML body onload that calls each ant to wiggle (e.g., antwiggle(0); antwiggle(1); (7.5)

Test the wiggle (1 mark)

Create the animation that causes the ant to crawl down. The animation should end if the ant is dead or it is game over (7.1, 7.3)

Link this to the function linked to the HTML body onload. So now you would have two more lines like antmove(0); antmove(1);

Test whether the ants walk (1 mark)

Create a function that kills an ant. What this should do is

* Check if the ant is alive. If it is not alive, do nothing. Otherwise, do everything else in the list.
* Set the ant is alive to false (3.13)
* Create an audio object linked to the squishing noise (2.14, 4.12)
* Play the squishing noise (7.6)
* Replace the image of the ant with a squished ant (2.13)
* The hidden form field stating the number of dead ants should increase by 1 (3.13, 3.22)
* The contents of scoreboard should say “Ants killed: ” and then the value of the number of dead ants (3.13, 3.23)

Create a button linked to the function that kills an ant to kill ant 0.

Test that ant 0 dies, but ant 1 does not.

Create an animation for the game over. What we do is:

* Create an audio object linked to the munching noise (2.14, 4.12)
* Play the munching noise (7.6)
* in JavaScript, create a span tag. (2.10, 4.12)
* The span tag has a z-Index of 1000 (style=”z-index: 1000”) (assignment instructions)
* The innerHTML of the span has the words “Game Over.” (3.24)
* The font size is initially 14 (7.12)
* and the span is located at coordinates (100,200) (2.26)
* Then, we animate the text. Every 20 clock ticks, the font-size increases by 2 until a font-size of 44 is reached (7.1)

In the function that causes the ants to move, state that if the ant is at a y position of 500 or more

* The hidden form field identifying whether it is game over is set to true
* The game over animation is displayed

Test if game over occurs if an ant reaches the picnic basket. (1 mark)

Create the function that detects if the mouse is clicked on an ant

Loop through all ants

If the mouse clientX is between the ant’s left and the ant’s left+30 and the mouse clientY is between the ant’s top and ant’s top+30 and the ant is alive

Call the function that kills the ant

Create a function that detects if the touch is on an ant. This is similar to the mouse function above.

Link the mouse function to the mouse event (7.8)

Link the touch function to the touch event (7.9)

Test mouse and touch (1 mark)

Create a function to create an ant

* The function reads the number of ants. (3.13)
* It creates an image field for the new ant, and hidden form fields for the x and y coordinates for the ant and the alive state of the ant, giving all of these the array index of the number of ants. (4.9, 4.12)
* It sets the alive state of the ant to 0, and the y coordinate of the ant to 0 (3.13)
* It sets the x coordinate of the ant to a random integer between 0 and 500 (7.11)
* It wiggles and moves the ant
* It increments the number of ants by 1 (3.22)

Set the number of ants to 0

Change the function linked to onload to create two ants

Test if this works (1 mark)

Create an animation function that does the following (7.1)

* In the outside function, declare a variable called difficulty and set it to 1 (3.12)
* In the outside function, declare a variable called counter and set it to 0 (3.12)
* In the outside function, declare a variable called spawner and set it to 0 (3.12)
* Set the animation to occur every clock tick. Yes, every clock tick (7.1)
* The animation ends when it is game over (7.1)
* In the inside function (the function that animates the frame), increment both counter and spawner (3.22)
* If counter=1000, difficulty is incremented by 1, and counter is set to 0 (3.12, 3.22)
* If spawner= the integer division of 1000/difficulty, spawn an ant and set spawner to 0 (3.12, 3.22)

The game’s onload function should link to this last function you wrote

Test the game (all marks)