Answer the questions in the spaces provided on the question sheets. If you run out of room for an answer, continue on the back of the page.

Name: _____

Overview

1. In today's lecture, we will implement a variation of the game hide and seek. Doing so will provide us with a test platform for learning.
   (a) How do the hiding locations get rendered on the level?

   (b) Determine whether or not the tile $(0, 6)$ is a hiding location. Print `true` or `false` for the result.

   (c) Given the `boy` avatar, determine which tile the `boy` is located in. Hint: This function is already in the system.

   (d) Give a tile location called `h`, determine the center (in pixels) of the tile.

   (e) Modify the `boy` blackboard so that his `hiding` property is `true`.

   (f) Imagine that you have a list called `looked`. This list contains all of the places that you have already searched. You have now searched $(3, 5)$. Add this location to the `looked` list.

(g) Select a random element from the list `x = [(3,5), (2,4), (1,1), (0,7)]`

Player Modifications

2. In this section, you will add some limitations to the player (that's you). This technique is often used for cut scenes in games.

   (a) If the player moves into a hidden location, stop responding their keyboard input. In addition, automatically move them (using their natural `speed`) to within 5 pixels (using `mandist`) of the center of the tile.

   (b) Set the `boy`'s `hiding` property to `true` in the blackboard.

Finite State Machines

3. We will now modify the AI (`girl`) so that it has three states: `WaitState`, for when the AI is waiting on the player to hide, `SeekState`, for when the AI is actively searching for the player, and `HomeState`, for when the AI is returning to her home location.

4. Create stubs for each of the three states. Don't forget to inherit from the parent `State` class.

5. For the `WaitState`:

   (a) Add `getEntryAction` logic to initialize the game. The avatar's status line should change to `Wait`. In addition, set the AI's blackboard so that `looked` is an empty list.

(b) We'll assume that the AI is not allowed to cheat (and therefore cannot peek to see where the player is hiding). In `getAction`, write logic so that the AI switches to the `SeekState` when the player's blackboard indicates that he is hiding.

6. For the `SeekState`:

(a) Add `getEntryAction` logic to set the status line to `Seek`. In addition, randomly select a hiding location to search that you have not already searched; store this location in `self.look`.

(b) For the `getAction`, use pathfinding to go toward the selected hidden location. Use a threshold of 5 to determine when you have arrived at the center of the tile. When you are in the same tile as the player, switch to the `HomeState`.

7. For the `HomeState`:

(a) For the `getAction`, return to the home location.

(b) For the `getExitAction`, reset the `boy`'s hiding property to `false`. In addition, teleport him to his home location.

You have now completed the test bed for learning. In the next lecture, your AI will intelligently find your hidden locations based on your past history.