# CS6310 A4 question_answers
**Yun Zhang / yzhang3426**

For each question, provide a relatively short (3-7 sentences) answer that identifies the specific classes, attributes, operations and other aspects of your design that allow you to address the given issues.

How do you track the current direction and location of each mower?

- direction: Mower - direction
- location: Mower - coordinates, Lawn - grid, MowerController - mowerSharedMap - grid
- In the beginning of the program, Mower coordinates are initialized to (-1, -1) to indicate unknown. During the first few turns of the simulation run, each mower will do a linear scan in four directions: north, east, south, west. Using the linear scan results, the mower will calculate the dimensions of the lawn and its own absolute coordinates on the lawn. The computed coordinates will be saved in the Mower - coordinates attribute.
- When a mower moves or gets removed from the lawn, its coordinates in the Mower object, Lawn - grid object and MowerController - mowerSharedMap - grid object will be updated accordingly.

How do you track the locations of gophers?

- Configuration - coordinatesOfGophers attribute stores the initial information parsed from the input configuration file, including the coordinates of the gophers.
- Lawn - Grid object stores the coordinates of all objects on the lawn including gophers.
- Gopher - coordinates attribute stores the gopher object's current absolute coordinates on the lawn.

How do you track of how much of the grass has been cut so far?

- component: Controller object - initGrassToCut(), Lawn - grassToCut.
- The Lawn object's grassToCut attribute tracks the number of grass to cut. The number of grass that have been cut can be computed from initGrassToCut() - grassToCut.

How do you update the simulation state for each of the varied actions?

- cscan() and lscan(): The scanned data will be temporarily saved in MowerController - mowerSharedMap - lastScanResult object. This scanned result will be incorporated into the mowerSharedMap - grid object by addScanResult(). Mower's batteryLevel will be updated by reducing the previous batteryLevel by the cost of this action.
- move(): The square the mower occupied on the global map (Controller - lawn - grid) and local map (MowerController - mowerSharedMap - grid) will be updated by removing mower in the content set. The mower's new coordinates will be calculated by adding previous coordinates to the mower direction's offset. If the new coordinates isOutOfBounds() or isOverGopher() or collidesWithMower(), the mower will be removed from the lawn, with its state set to killed and coordinate set to null. Otherwise, the new coordinates will be saved in Mower - coordinates. The new square the mower will

occupy on the global map (Controller - lawn - grid) and local map (MowerController - mowerSharedMap - grid) will be updated by adding mower in the content set. If the square has grass in the content set, grass will be removed from both maps and the lawn will be updated using updateGrassToCut(). Lastly, mower's batteryLevel will be updated by reducing the previous batteryLevel by the cost of this action.

- steer(): If the proposed direction is null or unknown, the mower will be removed from the lawn as described above. Mower's updated direction will be saved in Mower - direction attribute. Mower's batteryLevel will be updated by reducing the previous batteryLevel by the cost of this action.
- pass(): No state information needs to be updated except that the mower has performed action for this turn.

How do you determine the appropriate output for a cscan() action?

- components: MowerController - mowers, MowerController - lawn - grid - getCell()
- The MowerController has access to the mower's absolute coordinates and the lawn object. The coordinates of the 8 neighbors of the mower can be calculated by adding the mower's coordinates and each direction's offset. To get the ContentType of each direct neighbor of the mower, the MowerController object will look up the lawn grid with the corresponding coordinates.

How do you determine the appropriate output for an lscan() action?

- components: MowerController - mowers, MowerController - lawn - grid - getCell()
- The MowerController has access to the mower's absolute coordinates and the lawn object. The coordinates of the squares between the front of the mower and the first fence can be calculated by adding the mower's coordinates and the facing direction's offset. To get the ContentType of each requested square, the MowerController object will look up the lawn grid with the corresponding coordinates.

How do you determine when the simulation should be halted?

- component: Controller object - hasNextTurn().
- There are three scenarios that could terminate the simulation: all mowers have been killed by collision or gophers, all grass has been cut, or the number of turns has been exhausted. hasNextTurn() will check all these scenarios like so: hasLiveMowers() && lawn.getGrassToCut() > 0 && getTurnCounter() < maxNumOfTurns

How do you keep track of the knowledge needed to display the final report?

- lawn dimensions: Configuration - getLawnWidth(), getLawnLength().
- grass squares that originally existed on the lawn: Controller object - initGrassToCut(). This method computes the number using the lawn dimensions and the number of mowers and number of craters stored in the config object.
- grass squares that were cut: Controller object - initGrassToCut(), grassToCut. This value will be computed from these two values.
- number of completed turns: Configuration object - turnCount.

How do you keep track of the partial knowledge collected by each mower?

- component: MowerController - mowerSharedMap, MowerController - mowerSharedMap - lastScanResult
- Each mower learns sections of the lawn through scanning. The scan result is saved in the lastScanResult object.
- The partial knowledge learned by all mowers is merged into the mowerSharedMap data structure.

How do you manage collaboration between the mowers?

- Mowers share their partial knowledge of the lawn through the MowerController - mowerSharedMap object.
- The partial knowledge learned by all mowers is merged into mowerSharedMap.
- Each mower obtains the collective knowledge by accessing this mowerSharedMap object.

How do you determine the next action for a mower?

- components: Mower - chooseAction(), MowerController - mowerSharedMap
- Each mower has access to the accumulated knowledge from all mowers stored in MowerController - mowerSharedMap object. Each mower will use this information and its current energy level to decide what is the next action to perform. At the high level, as a simple example, if there is grass in front it and its energy level is not low, it could move to the next cell. If its energy level is low, it would move towards the closest charging pad. Otherwise, it would turn to the side with the maximum length of grass cells, based on the mowerSharedMap.