Rajesh Pothamsetty(rajeshpotham@gatech.edu)
User Id: rpothamsetty3
Student Id: 903154360
Assignment 2

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

    a. Class SimulationMonitor has two lists, one for storing positions and other for directions of lawn mowers.

    - lawnMowerPos:List<LawnMower, List<Integer>>
    - lawnMowerDir::List<LawnMower, Direction>

    For each turn in the SimulationMonitor, each lawn mower gets an action, for which the LawnMower class is used. Once the action is performed by the LawnMower object, lawnMowerPos and lawnMowerDir lists are updated using getter of LawnMower object. Initially, lawn mower positions are read in to a HashMap in the ReadScenario class which are used in initializing the LawnMower objects.

2. How do you track the locations of craters?

    a. Class ReadScenario, takes in the scenario filepath via SimulationMonitor class. ReadScenario reads the input file and creates a cratersPositions (HashMap) for craters as Key (integer id) and their X,Y co-ordinates as list integer value.

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

    a. Class SimulationMonitor tracks grass left to cut using grassLeft variable. getGrassCut() method which uses 'width', 'length', 'craters' getters of ReadScenario class to calculate TotalGrass and subtracts grassLeft to calculate grass cut so far.

4. How do you determine which mower will be selected to execute the next action?

    a. Until no lawn mower is crashed or removed, lawn mowers are selected in the round-robin fashion(in the order given in scenario file).

    - lawnMowerid (LawnMower class variable)

    Lawn mowers are stored in List of Lists as lawnMowers variable in SimulationMonitor class, in the same order given in the scenario file. lawnMowerid variables are also given ordinal integers.

    b. If a lawn mower is removed or crashed, isAlive() method of LawnMower class returns False, then we move on to the next LawnMower object in the lawnMowers list.

5. How do you validate that an action is valid?

    a. FinNextAction() method of LawnMower class returns an action

    a. Using the current state of the lawn and current mower position, algorithm will output an action.
    b. Naïve Ex: if all the 8 neighbor LawnSquare grass is cut, algorithm may decide to return "scan" action to find grass farther away.
    c. Naïve Ex2: If grass present in neighbor square, it may decide to cut and returns "move" action.
  b. ValidateMowerAction() method takes the input from FindNextAction() and validates it using the complete picture of the SimulationMonitor.
    a. If the output from FindNextAction is move, ValidateMowerAction checks if the mower may crash in to a CRATER or FENCE. cratersPositions variable of ReadScenario is used here.
    b. Ex: if this is the last turn available and the FindNextAction choses to steer or scan when grass is available in the neighboring square

6. How do you update the simulation state for move() and steer() actions?
  a. All the updates are done using the UpdateSimulationStatus()
    i. If move successful/fail(crash)
      1. update lawnMowerPos
      2. Update grassLeft
      3. Update lawnMap
      4. Update turnsLeft (if all mowers complete one action)
      5. Update liveLawnMowers (if crash)
    ii. If steer successful
      1. update lawnMowerDir
      2. Update turnsLeft (if all mowers complete one action)

7. How do you determine the appropriate output for a scan() action?
  a. Output for the scan() operation used by FindNextAction()
  b. Output of scan operation is a HashMap <Direction, LawnSquare>, i.e it has mapping of direction to square type ex: 'NortEast' maps to 'CRATER', 'East' maps to 'GRASS'.
  c. UpdateScanInfo() updates prevScanInfo variable based on the current action taken.
8. How do you determine when the simulation should be halted?
  a. haltSimulation() method of SimulationMonitor is executed after each action to check for
    i. if all mowers are not alive i.e liveLawnMowers = 0 => halt
    ii. turnsLeft = 0 => halt
    iii. grassLeft = 0 (getGrassCur()) => halt

9. How do you keep track of the knowledge needed to display the final report?
  a. All the data required for final report are stored as part of the variables in SimulationMonitor (lawnMap, turnsLeft, getLiveLawnMowers, grassleft, getGrassCut(), lawnMowerPos, lawnMowerDir etc.).
  b. All this information is displayed as report by the MowerStateSummary class
    i. This class uses getters of the above mentioned SimulationMonitor class.

10. How do you keep track of the partial knowledge collected by each mower?
    a. As discussed in the prev. questions
        i. lawnMowerPos
        ii. lawnMowerDir
        iii. cratePositions
        iv. lawnMap
        v. prevScanInfo for a specific mower

11. How do you determine which sections of the lawn still need to be cut?
    a. lawnMap which is a grid (LawnSquare[][]), maintains the full state of the grid in the entire simulation. If the squares of lawnMap has values enum type GRASS(1), then that square still needs to be cut.

12. How do you determine the next action for a mower?

    SimulationMonitor and LawnMower classes interact with each other, LawnMower class methods make use of the complete state of the lawn (lawnMap, crater positions etc.)

    FindNextAction() method returns the next action to be taken by a mower.

    - Using scan() method of LawnMower class
    - prevScanInfo variable of LawnMower class
    - using the above data, an intelligent algorithm will suggest an action which then needs to be validated.