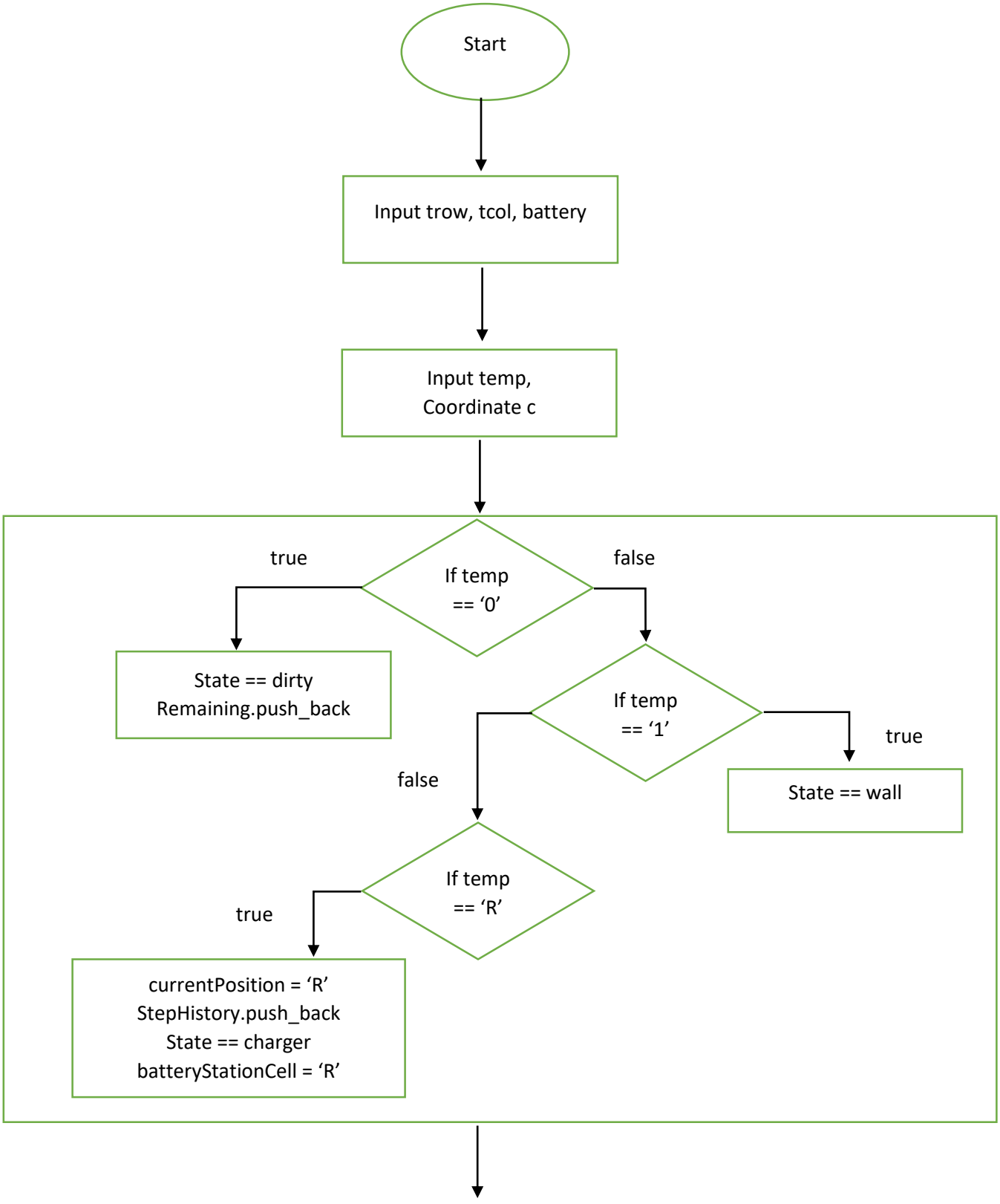
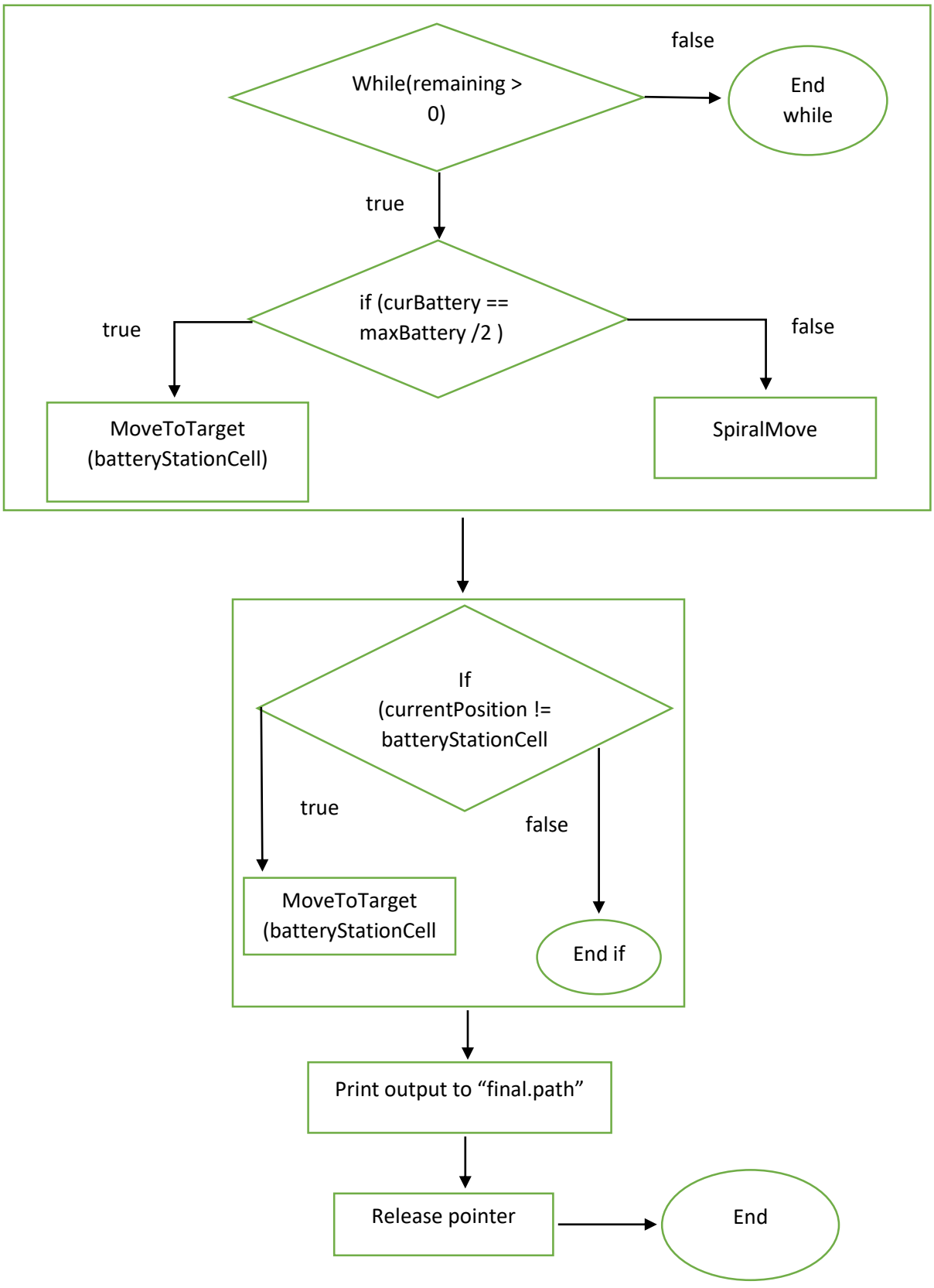
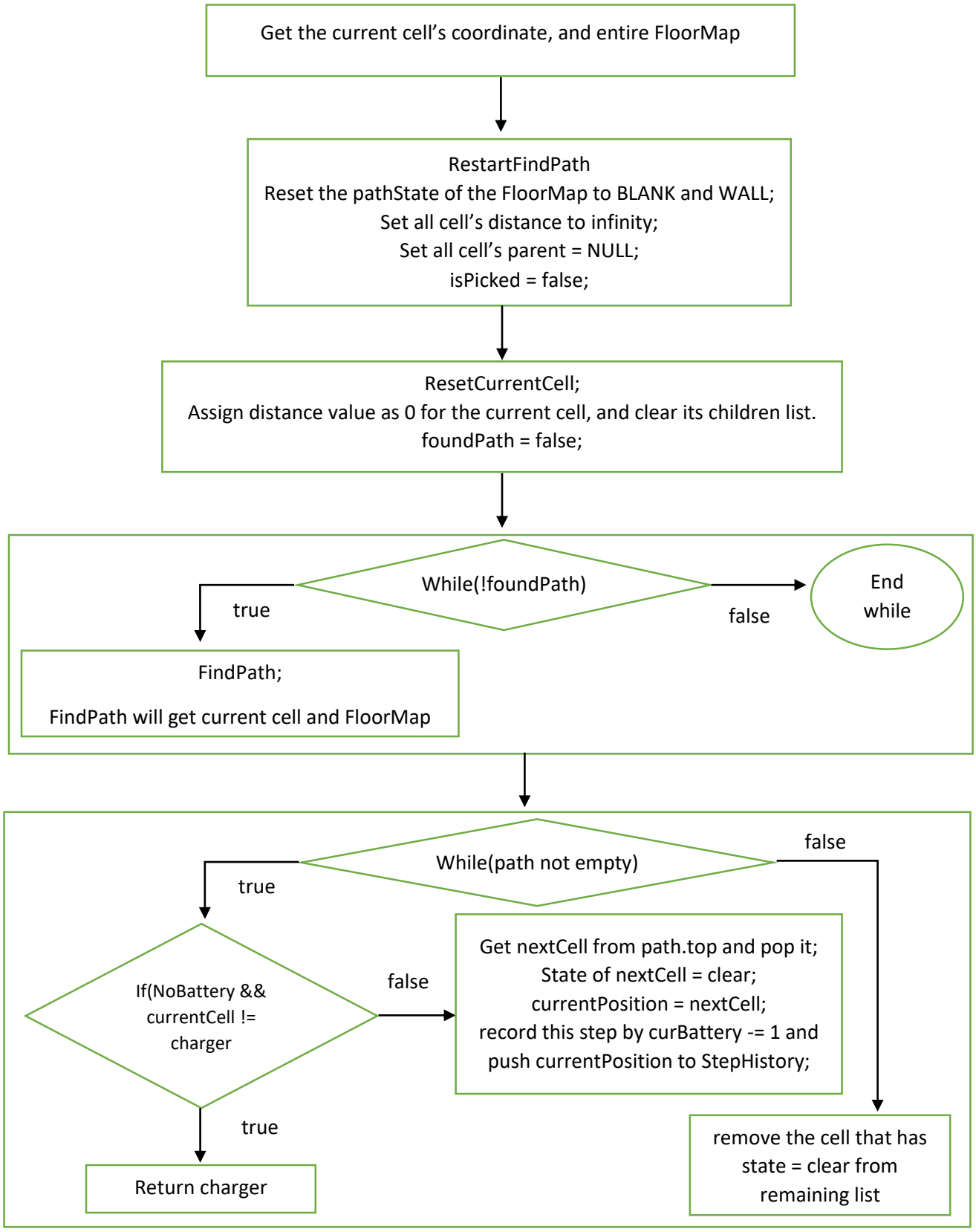
1. Program Flow Chart

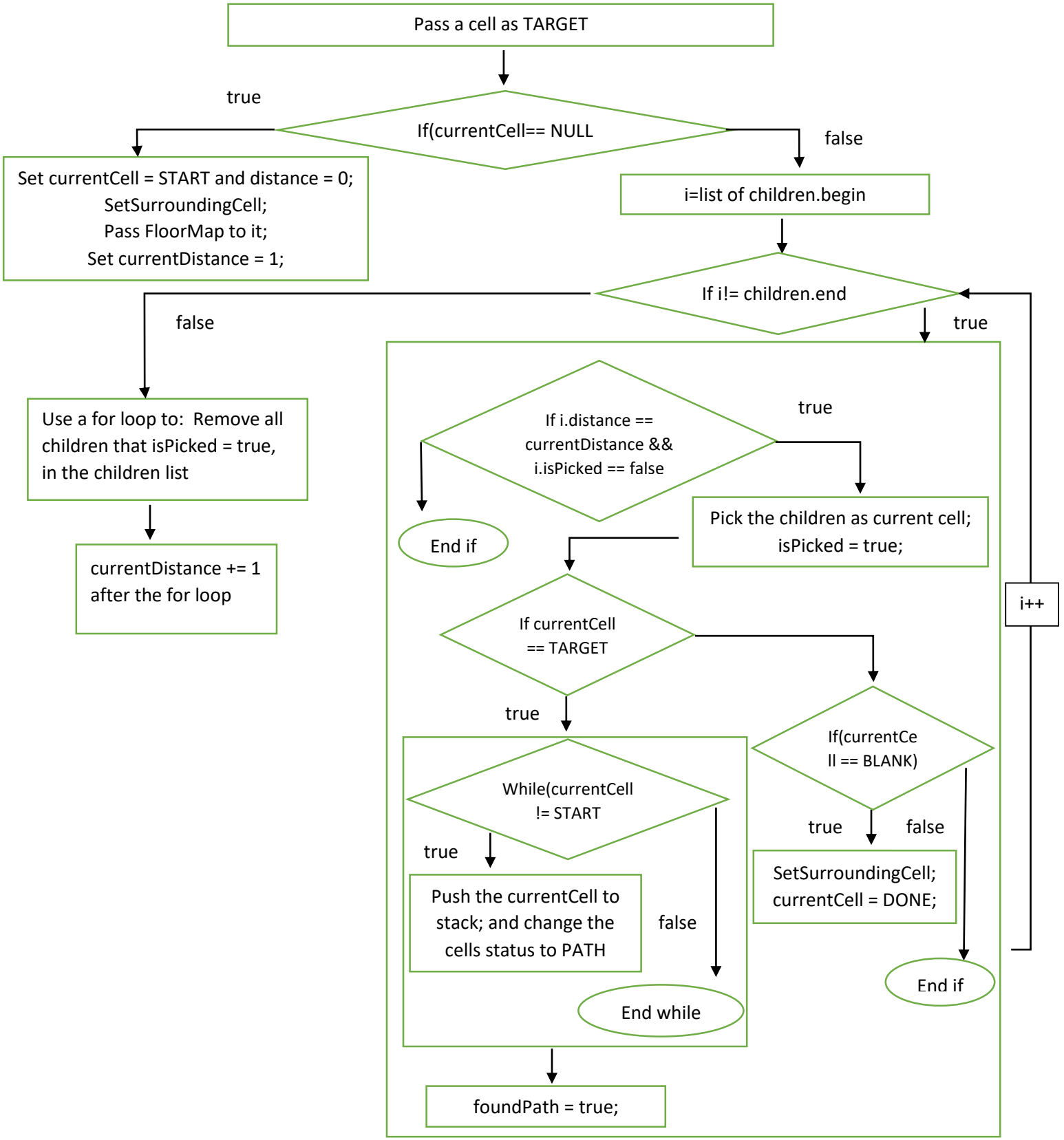



While(remaining > 0)
false
End while
true
if (curBattery == maxBattery /2)
true
false
MoveToTarget (batteryStationCell)
SpiralMove
If (currentPosition != batteryStationCell
true
false
MoveToTarget (batteryStationCell
End if
Print output to “final.path”
Release pointer
End

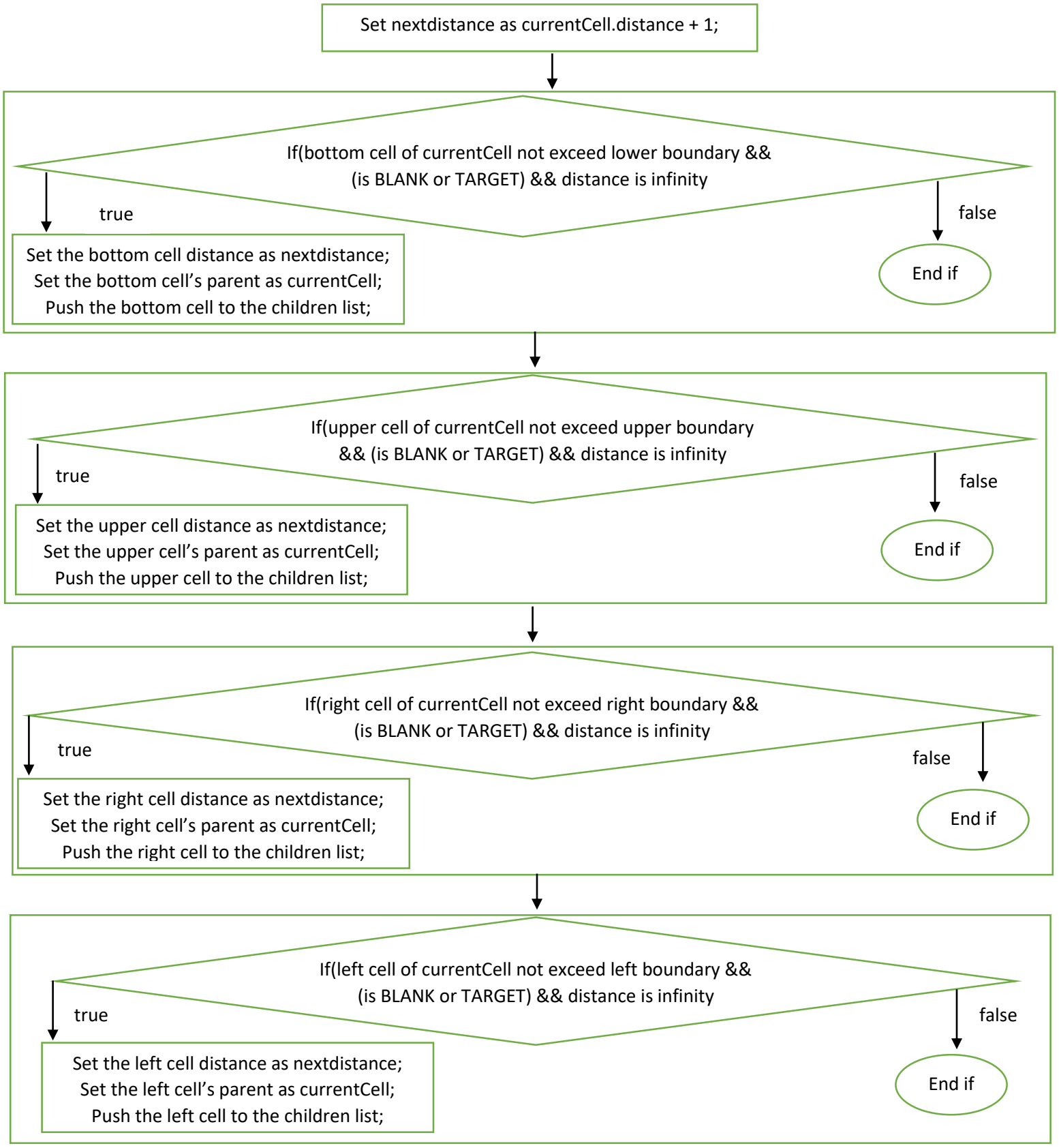
1.1 MoveToTarget Function



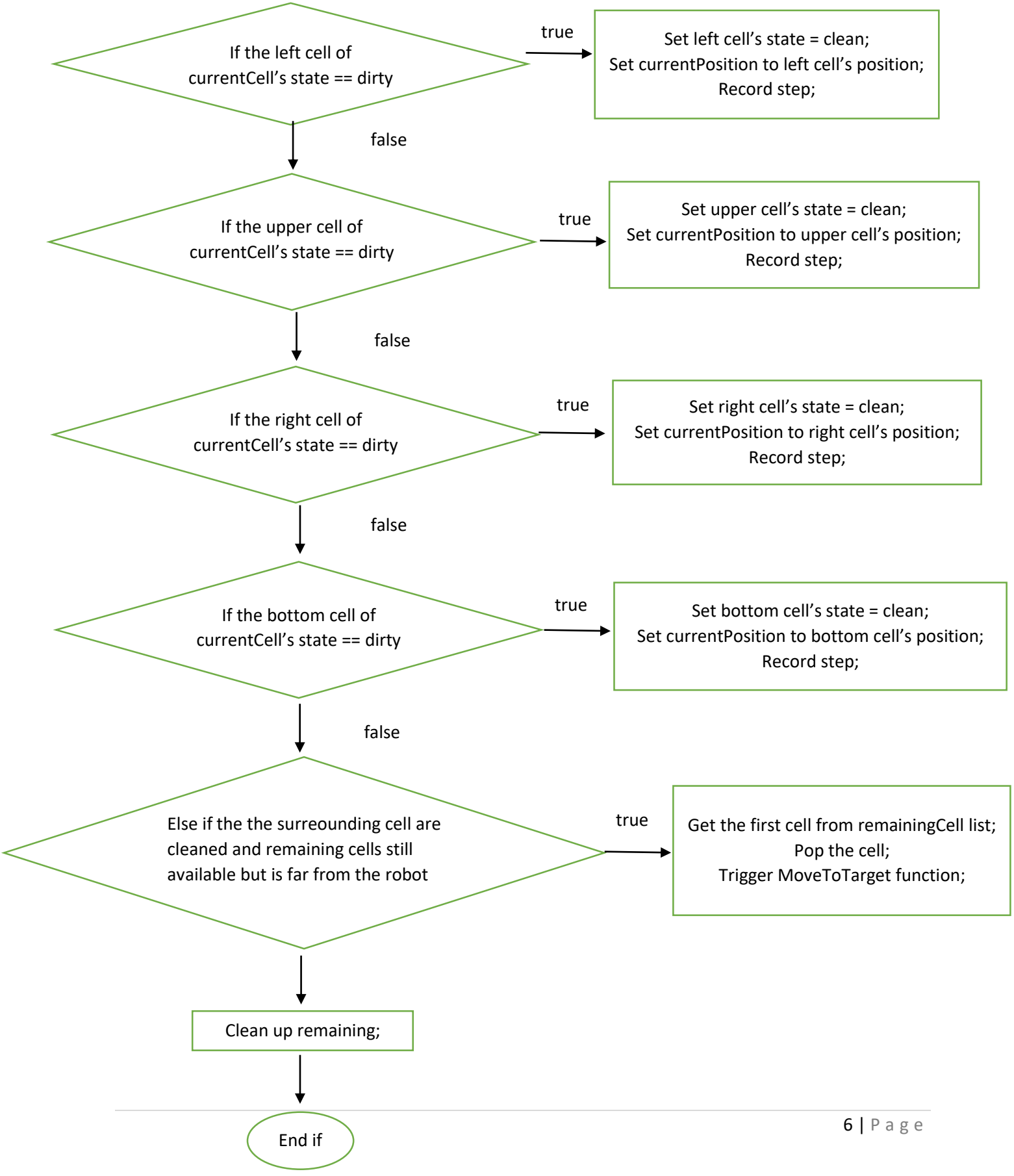
1.2 FindPath Function



1.3 SetSurroundingCell Function



1.4 SpiralMove Function



2. Detailed Description

The row, the column and the battery are read from the file and and call the FloorMap function to build a trow * tcol 2d array (floor), the data type of floor is “Cell” class. The state of each input will be recorded, ‘0’ as ‘dirty’; ‘1’ as ‘wall’, ‘R’ as ‘charger’. Every ‘0’ read, it will be push to the remainingCell list. Then, the position of ‘R’ will be set as current position, and push to the vector StepHistory, with coordinate x and y. A variable called batteryStationCell is use to record the coordinate of the charger.

While the remainingCell list is not empty, check if the battery still enough to continue for cleaning, if not, go to batteryStationCell to charge until full, and continue cleaning process; else continue cleaning process until battery not enough or until there is no any remainingCell.

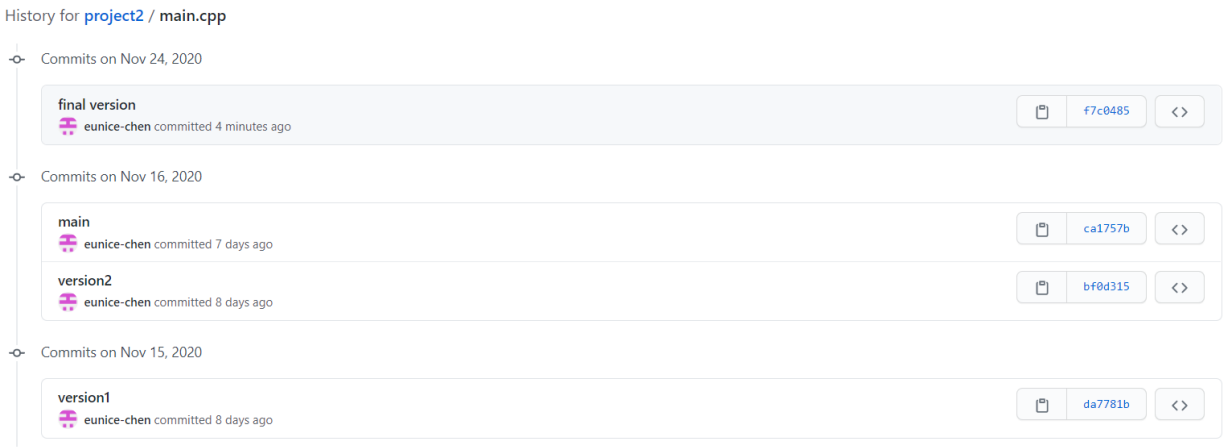
The SpiralMove function is to let robot move in a spiral shape in the map until there are no any dirty cells nearby or reached the wall or boundary, then the robot will move to another direction perform same action. The directions for the robot to move are left, up, right and down. When the robot cleans up the cell, the step will be recorded. If the surrounding cell of the robot is cleaned, the robot will trigger MoveToTarget function.

The MoveToTarget function uses Dijkstra’s algorithm. All the cells distances (vertices distances) are set to infinity, and the current position distance (source distance) are set to 0. Set a target cell, then find a path to the target cell. The way to find path is starting with setting currentCell pointer to current position as it is null. Then, a function named SetSurroundingCell is called. In SetSurroundingCell function, start with set the variable of nextDistance to current cell’s distance +1. If the surrounding cell of robot is in the boundary, and not reached before, or is target cell, and its distance is infinity, the cell’s distance will be set to nextDistance, and will be push to the children list, and its parent is current cell. Next step will be setting current distance of robot or path finding to 1 for the coming children list. Check children list for possible target cell, if target cell not found, iterate SetSurroundingCell function and check the newly added children until the target cell is found. The children cell also will be erase once they are checked if target is not found. Update current distance after all children are filtered. Once target cell is found, the robot will start moving according to the path founded. The robot will keep moving until battery is not enough, and will go back to batteryStationCell for charge to max battery and continue to move. For every cell in path that is visited, the path cell will be deleted and the step will be recorded. For every remainingCell that is cleaned will be removed from the list.

In the end the result will be the steps of the move and every coordinate will be written to the file.

3. Detailed Description of the Test Case

My test case is to test when the ‘wall’ is not irregular, the robot can still perform to clean all the ‘dirty’ cells.

Git Hub

https://github.com/eunice-chen/project2/blob/master/main.cpp