## HW 4 - Robot Vacuum Cleaner\_COMP220

## Note

Please note that this is a long-term assignment. It is not due until the day before your final exam. But... Please do not wait until the end to begin work on it. You can pursue this project using any approach you learn from our class, or any inspiration you have from other sources. It is supposed to be a more open-ended and explorative exercise, though some of our discussions later in the semester can prove very relevant to its most optimal implementation.

HINT: I would recommend that you first implement a "random" solution that attempts to accomplish the goal through random chance (while adhering to the rules below). Once you have that part implemented, you may be more oriented to the nature of the problem and how to resolving it efficiently.

## Overview

Design and implement a basic algorithm for an automated robot that must clean a rectangular area that is populated with random obstacles. Your robot is so powerful that it can sweep up both garbage and obstacles, but you do not want to sweep up obstacles if you want to receive full credit. Your robot can only move 1 square at a time in any direction (including diagonally), but only using the **moveTo**() function provide.

## Getting Started

Use code base at <https://repl.it/@cstutormarin/cs220HWstarterrobotvacuum>

[(Links to an external site.)](https://repl.it/@cstutormarin/cs220HWstarterrobotvacuum)

## Background

A grid is filled with garbage and obstacles. Your robot is so powerful that it can sweep up both! But you only want to sweep up the garbage. Our "room" is just a square grid represented by a 2D grid array of single characters. A piece of garbage is denoted by the period character '.' (note we use single quotes in cpp for characters!). An obstacle is denoted by a the star character '\*'. Your grid goes from row zero to row **numrows**-1 and from column zero to **numcols**-1. Your robot can only move 1 square at any time in any direction (it can move diagonally). But be careful not to move out of the valid range of the grid - you could crash your program.

## Details

You need to implement ONLY the **sweepGrid**() function so that after being called in the main program, the global **grid**[][] array will have been cleaned of all of its garbage, but none of the original obstacles should have been removed. A scoring function will be called to compare your final grid to the original grid to see if you have destroyed any obstacles and how much garbage you have swept up.

Your **sweepGrid**() routine MUST use the **getRow**() and **getCol**() functions to determine the current location of your robot in the grid at any time. (see examples in code)

Your **sweepGrid**() routine must also use the **moveTo**() function to **ATTEMPT** to move your robot to a specific desired location, where it will sweep up the item at that location (be it garbage or an obstacle). The **moveTo**() function will return true if the move was successful, and false otherwise. False will be returned if you attempt to move more than 1 space in any direction at one time. Or if you attempt to move out of the grid (which would crash the program). The moveTo() function will sweep up what ever is at the location you tell the robot to move into. The **moveTo**() function will also keep a count of your moves to help with scoring.

Your room will be filled with 25% randomized obstacles (stars), and every other location of the grid will be filled with garbage (dots).

You should not return from the **sweepGrid**() function until you believe the entire room is cleaned of garbage, or until you give up.

## Global Variables

These are the key variables you will want to use in your sweepGrid() routine:

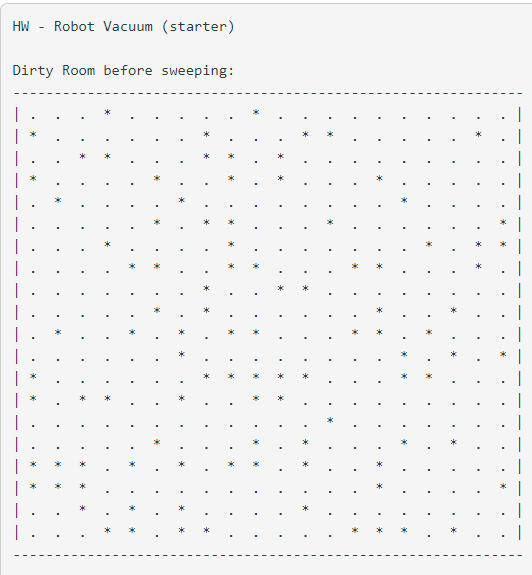
* const int **numcols** = 20; // number of columns in our grid (width)
* const int **numrows** = 20; // number of rows in our grid (height)
* char **grid**[numrows][numcols]; // declare our grid
* char **original**[numrows][numcols]; // declare our a duplicate of our grid for reference
* int **currentRow** = 0; // starting row location of robot
* int **currentCol** = 0; // starting col location of robot
* int **totalMoves** = 0; // total count of moves made by the robot

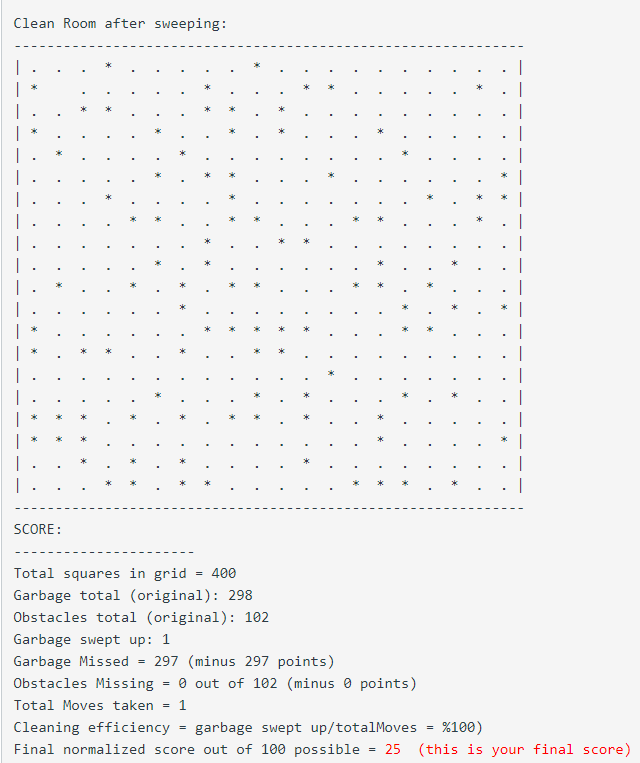
## Functions you should call

These are the functions already written for you that you can call from your **sweepGrid**() function:

* int **getRow**(); // return the current row location of the robot
* int **getCol**(); // return the current row location of the robot
* bool **moveTo**(int r, int c); // move the robot to this location and sweep up what is there

## Example Output





## **Testing**

Run the program at the link above to see your preliminary score (out of 100 possible points). An efficiency rating will also be printed which will be taken into consideration for a bonus assessment (see more below).

## **Submission**

Submit your entire running program via repl or plane text file. (please don't just send me your sweep function, even through that is all you will be writing.

## **Scoring (100 points total)**

* **100 points**: based entirely on the score printed by the program and based on your ability to clean all garbage without sweeping up obstacles.

## **Bonus(+25 points max)**

* **+5 points for every +5% level of efficiency you achieve above 75%**
  + <75% efficiency - no bonus points
  + < 80% efficiency = +5 points
  + <85% efficiency = +10 points
  + <90% efficiency = +15 points
  + <95% efficiency = +20 points
  + <=100 efficiency = +25 points