Joseph Baruch

CS 121

Professor: Bruce Bolden

Program 2

Due: 2/10/2023

Program Design

Objective

The objective of this assignment and program is to become more familiar with dynamically allocated arrays and work through the problems in this program in an organized, efficient, and proper way.

Program Description

Program 2 is based around an idea where a Roomba will move around a room and will try to cover the entire floor space. Translating this into C++ code, we will use a dynamically allocated array as the floor and a current position of the Roomba as a value of rows and columns. After starting at a set initial position, the program will run until the “Roomba” has randomly visited every cell or the allotted maximum number of moves has been met.

After the program terminates, print the following:

* Cell row and column values that has been visited the most.
* The number of times that cell has been visited.
* The number of moves made.
* The array with each cell and the number of times each cell has been visited.

Requirements

* Read the array row and column sizes.
* Read initial position.
* Read and integrate maximum number of moves allowed into the code.
* Dynamically Allocate a 2D Array using set row and column sizes.
* Randomly move the “Roomba” around the 2D array without going out of bounds or making the next move back into the current position.
* Display the required outputs (listed above) after the program is finished.
* Outputs: Proof the program worked of the following conditions.
  + nRows=15, nCols=10, row=7, col=4, k=250
  + nRows=10, nCols=15, row=1, col=1, k=2500
  + nRows=55, nCols=50, row=1, col=1, k=2500
  + Personal Test Case:
    - nRows=5, nCols=5, row=3, col=3, k=150

Program Contents

* Libraries
  + Iostream: Included for all the basic operations.
  + Ctime: Included for use in srand() to be utilized for pseudorandom Roomba move.
  + Iomanip: Include to allow use of setw( int w) in array print output.
* Functions
  + Main: Main operation place for the program. Described more within program operation description below.
  + Constructor: Initializes the values of all data types within class and dynamically allocates 2D Array.
  + Initialize: Sets all cells in the 2D array to equal 0.
  + UpdateBoard: Takes current array position (determined by row and col) and increases by 1. Also, increment move counter (moves) by 1.
  + RandomMove: Randomly moves current position to any of the eight surrounding cells. Uses a do-while loop to make sure the next move does not move off array. Within do-while loop there is a while loop that checks to make sure the next move isn’t back into its current cell.
  + MoveCheck: Checks array for any zeros. If there are no zeros, then program will print out final outputs. If there are zeros the program will continue.
  + MaxCel: Finds array cell with the greatest number of visits and the visit amount.
  + PrintArray: Prints array with correct cell width size for formatting purposes.
  + OutPrint: Prints the required outputs mentioned above in Program Description.
  + ClearMem: Deallocates 2D array.
* Class Roomba
  + Private Members:
    - Int \*\*TwoArray: Dynamically Allocated 2D Array.
    - Int nRows: Array rows.
    - Int nCols: Array columns.
    - Int row: Array row initial position.
    - Int col: Array column initial position.
    - Int max: Amount of visits the array cell that received the greatest number of visits had.
    - Int celwidth: Array cell width for formatting purposes.
  + Public Members:
    - Int k: Maximum number of moves allotted.
    - Int moves: Number of moves counted.
    - Roomba(): All functions described above.
    - void Initialize():
    - void UpdateBoard():
    - void RandomMove():
    - int MoveCheck():
    - void MaxCel():
    - void OutPrint():
    - void PrintArray():
    - void ClearMem():

Program Operation Description

1. Declare class.
2. Initialize 2D array to all “0”’s.
3. Update 2D array by adding a 1 to the initial position within the 2D Array.
4. Do-while loop that runs while allowed number of moves is greater than the current amount of moves and there is still “0”’s in the 2D array. Operations with do-while:
   1. Randomly move the current position in the 2D array to any of the eight surrounding cells. Check to make sure next move isn’t back to current location and check to make sure next move doesn’t go off 2D array.
   2. Update the 2D array by adding a 1 to the current location.
   3. Check the 2D array for any zeros.
5. Find the cell that was visited the maximum number of times and the amount it was visited.
6. Print the required outputs listed above.
7. Deallocated the memory for the 2D Array.

Time Estimation: ~ 14 hours

* Program Design: ~ 1.5 hours
* Programming Log: ~ .5 hours
* Program Code Frame: ~ 1 hour
* Finer Details: ~ 5 hours
* Researching and figuring out unknown problems: ~ 3.5 hours
* Testing: ~ .5 hours
* Commenting Program: ~ 1 hour
* Final Check: ~ .5 hours
* Putting together Final Product: ~ .5 hours

Programming Log

2/2/2023: 15 minutes

* Read programming assignment guidelines.

2/5/2023: 1.5 hours

* Reread programming assignment guidelines.
* Began Programming Log.
* Began Program Design.
* Experimented with coding classes and passing values.
* Started outline of program in visual studio code.
* Began to realize issues with how to do 2D arrays with dynamic allocation.

2/6/2023: 2 hours

* Got the basic outline finished and started filling my functions with everything I could.
* Ran into the issue again with not knowing how to make a 2D Dynamically allocated array. This resulted in my not getting very far into the program completion.
* Remembered how to build classes and finished implementing it.
* Decided to spend a big portion of my day in the CSAC on Tuesday because I needed help.

2/7/2023: 4 hours

* Finished all code on top of the basic outline of the code so I could go into tutoring prepared.
* Found out how to allocate 2D arrays dynamically from the tutor and Boldens lecture notes.
* Ran into issues with my for-loops and 2D arrays. I was getting a lot of segmentation faults because the for-loops were moving outside the bounds of the 2D array. I adjusted the values within the for loops and it fix the problem over my test cycles.
* I initially thought that problem was with the do-while loop but it was really because of the bounds problem.

2/8/2023: 4 hours

* I started to have an issue where sometimes my code would work and sometimes it wouldn’t. After spending half the day trying to figure out the issue by staring at it, scanning every line, and making sure every function worked properly I found it. The issue was, whenever the nRows > nCols the program would not run. The other way around would work though. I didn’t know what to do about it, so I emailed Professor Bolden. He said I had my rows and column backwards, but I was still confused. I went into his office hours the next day and figured it out.
* Ended the day with all my code finished and working properly except for the one major issue described above.

2/9/2023: 5 hours

* Went into office hours and learned how to use setw() which helped me solve the problem with my output format looking unformatted.
* Also found out from Professor Bolden that the dynamically allocated 2D array process was wrong. I had my rows and column mixed up. This fixed the major issue of when my rows > columns, I received a segmentation fault.
* Moved my outputs into a single function to clean up main after a comment from Professor Bolden.
* Finished Program Design by looking back at the comments from program 1.
* Finished Programming Log.
* Paste outputs and code into word document.
* Proofread everything.
* Printed out and stapled finished copy

2/10/2023: 5 minutes

* Turn in printed assignment to Prof Bolden in class at 9:30am.

Outputs

Note: I used setw() inside of my code to make sure that the formatting would look nice and it does when the compiler output’s it. When I paste the output into this word document it messes up the formatting. Let me know if you would like screenshot of the output from the compiler.

**nRows=15, nCols=10, row=7, col=4, k=250:**

joseph.baruch@Josephs-Air Programs % g++ program2.cpp

joseph.baruch@Josephs-Air Programs % ./a.out

Results:

Number of Moves: 250

Array:

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 1 2 1 2 1

0 0 0 2 2 2 0 1 1 0

0 0 0 0 2 0 0 0 1 1

0 0 0 0 0 3 1 1 1 1

0 0 0 0 0 4 7 5 3 0

0 0 1 0 2 1 3 3 2 3

2 5 2 2 2 2 5 3 9 5

3 1 6 8 5 7 6 4 3 4

1 5 7 6 9 10 5 5 7 4

2 5 5 7 8 6 4 5 3 0

Cell [6,14] was visited the most number of times.

It was visited 10 times.

**nRows=10, nCols=15, row=1, col=1, k=2500**

joseph.baruch@Josephs-Air Programs % g++ program2.cpp

joseph.baruch@Josephs-Air Programs % ./a.out

Results:

Number of Moves: 1224

Array:

11 15 8 5 7 6 5 7 5 9 6 1 3 3 1

3 11 7 9 5 10 13 5 11 10 12 4 2 5 3

6 6 13 10 8 6 10 13 6 12 7 4 2 6 7

3 9 9 8 4 5 8 7 11 15 13 8 6 9 7

2 7 6 4 5 9 12 8 13 8 11 13 15 19 5

2 3 2 5 4 6 10 15 5 9 12 13 10 18 6

2 7 5 5 6 3 12 11 8 6 11 18 15 4 8

10 12 7 5 6 11 7 8 6 6 13 16 13 18 8

4 13 10 3 9 14 11 12 5 8 12 14 14 11 9

2 3 5 9 10 9 14 6 6 5 12 14 8 6 2

Cell [14,5] was visited the most number of times.

It was visited 19 times.

**nRows=55, nCols=50, row=1, col=1, k=2500**

joseph.baruch@Josephs-Air Programs % g++ program2.cpp

joseph.baruch@Josephs-Air Programs % ./a.out

Results:

Number of Moves: 2500

Array:

3 3 1 1 2 1 0 3 2 0 1 0 0 0 0 0 0 0 0 1 2 3 3 11 10 8 2 6 4 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 1 2 5 2 1 3 2 3 3 2 2 2 0 0 0 0 1 1 2 4 6 11 9 8 1 6 4 2 1 0 0 2 4 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 5 2 1 1 1 1 0 1 4 9 12 4 2 2 2 2 1 1 1 5 10 12 8 3 4 1 1 2 0 0 1 4 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 0 2 3 2 1 0 0 3 6 8 5 6 4 7 5 5 5 2 4 5 6 5 6 5 0 0 0 1 1 3 4 4 4 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 3 4 2 1 1 1 0 5 7 6 7 4 6 6 6 4 5 4 3 3 4 2 0 1 0 1 1 3 6 6 2 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 3 1 0 0 0 0 2 2 3 5 6 10 4 5 4 2 1 2 2 2 5 3 1 0 4 3 10 6 5 3 8 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 1 0 0 1 0 0 1 6 7 3 4 4 2 5 5 3 9 4 3 5 0 3 4 9 6 6 2 8 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 1 1 2 2 1 1 4 4 5 7 2 1 2 2 6 10 6 12 5 4 5 7 6 3 3 4 5 5 5 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 2 4 4 3 6 6 4 1 0 3 2 3 3 9 6 6 7 7 5 2 5 7 4 2 3 5 3 3 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 1 2 2 1 5 6 9 4 2 5 5 5 4 2 1 1 8 4 3 4 4 1 5 5 4 1 4 0 4 3 7 2 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0

1 2 3 4 4 4 2 12 8 4 3 5 3 4 1 2 3 3 1 2 4 3 2 2 3 3 2 3 1 2 2 5 3 4 2 4 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 2 1 1 2 4 2 2 0 0 0 6 2 3 1 3 4 4 5 4 4 3 1 1 0 1 1 0 1 3 4 1 1 4 3 1 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 1 1 2 5 3 4 2 2 2 4 1 3 5 4 6 3 6 8 0 2 3 5 0 2 2 0 0 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 2 0 0 0 0 0 3 4 2 2 2 3 2 4 2 4 2 3 8 4 3 5 3 6 3 4 1 3 0 0 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

2 1 3 0 0 0 0 0 3 2 2 3 1 3 4 2 3 6 7 6 7 4 2 4 5 3 0 3 1 1 1 1 1 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 4 3 1 1 1 0 3 1 0 0 1 3 6 2 6 7 8 3 2 1 2 1 4 3 1 0 0 3 1 0 0 1 1 4 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

1 2 4 5 1 1 2 1 1 0 1 1 2 2 4 3 7 6 1 0 1 2 3 3 1 0 1 1 0 0 0 0 0 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 1 1 1 3 0 1 0 0 2 1 0 1 0 4 5 6 2 4 2 2 9 10 1 1 1 0 1 1 0 2 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 1 0 1 0 0 2 1 2 1 0 2 0 4 4 6 6 5 7 9 5 3 0 0 1 0 1 1 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 2 6 5 5 3 1 0 1 3 3 6 4 2 4 4 1 3 1 1 1 0 1 2 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 1 3 4 4 0 2 2 1 2 2 1 6 9 1 3 2 0 0 1 0 0 1 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 2 0 0 0 0 1 0 1 6 7 4 5 2 3 2 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 5 5 7 3 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 5 6 1 4 0 0 2 0 1 4 5 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 1 0 1 1 0 0 3 2 2 4 1 2 2 0 0 1 3 1 2 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 2 0 0 1 2 0 2 7 4 7 1 0 0 1 1 0 2 3 3 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 1 4 3 3 1 2 3 6 18 12 10 5 4 5 3 6 6 4 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 3 4 6 4 7 7 5 4 7 15 6 3 3 0 7 7 6 2 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 2 8 8 9 8 7 6 5 5 1 6 5 2 1 5 7 1 3 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 4 4 9 7 4 3 2 2 1 3 2 8 5 3 2 3 1 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 2 5 4 3 4 4 3 2 2 2 0 2 8 7 5 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 2 3 2 2 3 3 3 5 1 1 2 2 2 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 1 1 1 2 1 1 1 3 1 1 0 3 2 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 2 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Cell [19,27] was visited the most number of times.

It was visited 18 times.

**nRows=5, nCols=5, row=3, col=3, k=150**

joseph.baruch@Josephs-Air Programs % g++ program2.cpp

joseph.baruch@Josephs-Air Programs % ./a.out

Results:

Number of Moves: 150

Array:

0 1 3 8 4

4 5 7 8 12

3 7 8 16 3

4 8 7 5 4

4 10 11 7 1

Cell [4,3] was visited the most number of times.

It was visited 16 times.

Program

/\* Joseph Baruch

\* CS 121

\* Professor: Bruce Bolden

\* Program 2

\* Due: 2/10/2023

\*/

#include <iostream>

#include <ctime>

#include <iomanip>

using namespace std;

class roomba{

private:

int \*\*TwoArray; // Dynamically allocated 2D array.

int nRows, nCols, row, col, max, celwidth; // Defined in constructor.

public:

int k, moves; // Defined in constructor.

roomba(); // Constructor.

// ---- Function Prototypes -----

// Descriptions in main and within functions.

void Initialize();

void UpdateBoard();

void RandomMove();

int MoveCheck();

void MaxCel();

void OutPrint();

void PrintArray();

void ClearMem();

};

int main(){

srand(time(0));

roomba r;

int noZero;

r.Initialize(); // Set TwoArray[][] to all "0"'s.

r.UpdateBoard(); // Updates board (adds 1) to initial position.

do{ // Loops if allowed moves isn't met or 0's in array.

r.RandomMove(); // Randomly chooses move. Updates row and col.

r.UpdateBoard(); // Increase moves and current location in array by 1.

noZero = r.MoveCheck(); // Returns 0 if 0's in array. Returns 1 if no 0's.

}while(r.k > r.moves && noZero == 0);

r.MaxCel(); // Finds cell with most times visited at visit amount.

r.OutPrint(); // Print required outputs

r.ClearMem(); // Deallocates dynamically allocated 2D array.

}

roomba::roomba(){

nRows = 5; // Array rows.

nCols = 5; // Array columns.

// row and col initial position.

// After initial UpdateBoard() turns into current position.

row = 3;

col = 3;

moves = 0; // Number of moves.

k = 150; // Maximum amount of moves.

// ---------- Required Output Configurations -----------

// nRows=15, nCols=10, row=7, col=4, k=250 (passed)

// nRows=10, nCols=15, row=1, col=1, k=2500 (passed)

// nRows=55, nCols=50, row=1, col=1, k=2500 (passed)

// nRows=5 , nCols=5 , row=3, col=3, k=150 (passed)

celwidth = 3; // Array cell width (for formatting).

// Allocates TwoArray (2D Array).

TwoArray = new int\*[nRows];

for(int i = 0; i < nRows; i++){

TwoArray[i] = new int[nCols];

}

}

void roomba::Initialize(){

for(int i = 0; i < nRows; i++){

for(int j = 0; j < nCols; j++){

TwoArray[i][j] = 0; // Sets current position to "0".

}

}

}

void roomba::UpdateBoard(){

TwoArray[row-1][col-1] += 1;

moves += 1; // Increments moves.

}

void roomba::RandomMove(){

int moverow; // Row move amount.

int movecol; // Column move amount.

do{

// Sets movecol and moverow to either -1, 0, or, 1 (random).

movecol = rand() % 3 - 1;

moverow = rand() % 3 - 1;

// Makes sure move isn't back to current location.

while(movecol == 0 && movecol == 0){

movecol = rand() % 3 - 1;

moverow = rand() % 3 - 1;

}

// While: makes sure move doesn't go off the board.

}while((row + moverow) <= 0 || (row + moverow) > nRows || (col + movecol) <= 0 || (col + movecol) > nCols );

// If moverow and movecol is correct: create new current location.

row += moverow;

col += movecol;

}

int roomba::MoveCheck(){

for(int i = 0; i < nRows; i++){

for(int j = 0; j < nCols; j++){

if(TwoArray[i][j] == 0){

return 0; // If there are 0's: return 0.

}

}

}

return 1; // If there are NO 0's: return 1;

}

void roomba::MaxCel(){

max = 0; // Initializes max = 0.

for(int i = 0; i < nRows; i++){

for(int j = 0; j < nCols; j++){

// If TwoArray value is greater max: change max, row and col.

if(TwoArray[i][j] > max){

max = TwoArray[i][j];

row = i + 1;

col = j + 1;

}

}

}

}

void roomba::PrintArray(){

for(int i = 0; i < nRows; i++){

for(int j = 0; j < nCols; j++){

// Prints TwoArray's current position in for loops.

cout << setw(celwidth) << TwoArray[i][j];

}

cout << endl;

}

}

void roomba::OutPrint(){

cout << "Results: " << endl << endl;

cout << "Number of Moves: " << moves << endl << endl;

cout << "Array:" << endl;

PrintArray(); // Prints array.

cout << "\nCell [" << col << "," << row << "] was visited the most number of times." << endl;

cout << "It was visited " << max << " times." << endl;

}

void roomba::ClearMem(){

// Deallocates TwoArray

for( int i = 0 ; i < nRows ; i++ ){

delete TwoArray[i];

}

delete [] TwoArray;

}