Offline Questions:

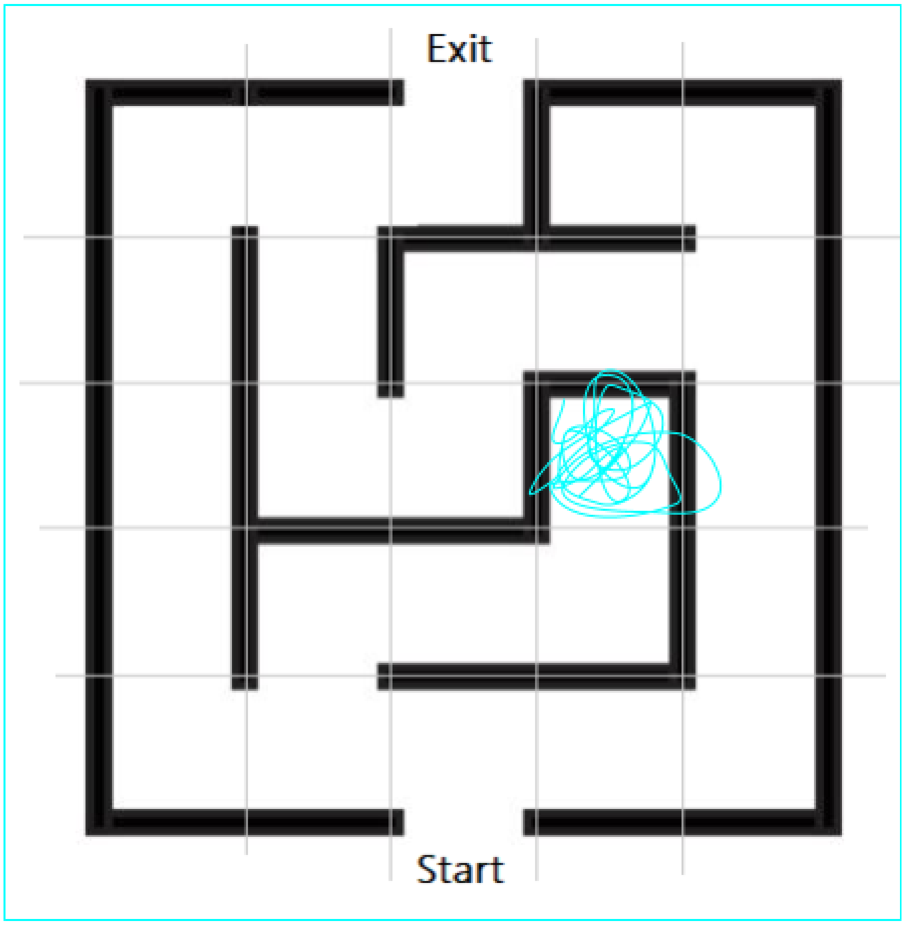
In the following maze, how many squares would you touch following the 'right wall' trick?

17 squares

How many different segments of wall would you touch? (If you backtrack, that counts as extra squares touched!)

17 Segments

The 'right wall' trick is a method of solving certain mazes by keeping your hand on the 'right wall'. This method eventually guarantees finding an exit.



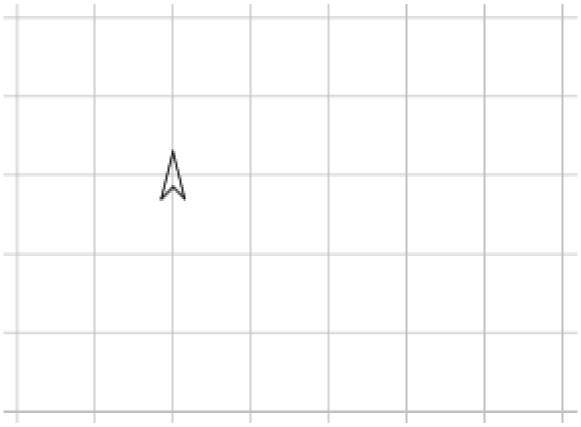
The right wall strategy is a guaranteed way to get through a maze if you start at the entrance. But what if you woke up in the middle of the maze?

Find a square in the same maze where if you tried the right wall trick, it would NOT get you to either the exit or the entrance (no matter which wall you choose from that square)!

The square is scribbled in blue.

Analyzing an Algorithm

Carol the robot moves along the lines of a grid. Here, we denote her position with an arrow whose tip points to the direction in which she moves. (This is just for diagramming purposes. You'll see later how she actually looks.)
She always moves in the counterclockwise direction along the wall, and her initial direction is north as indicated in the diagram.

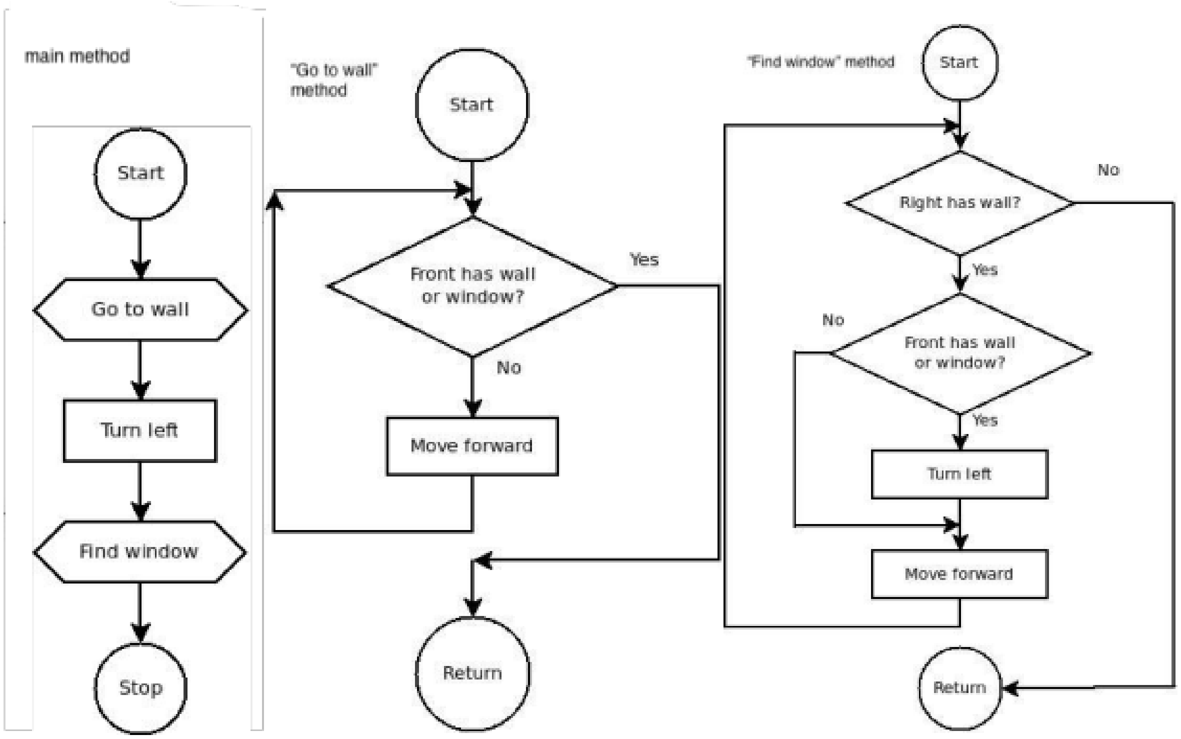


She can perform the following five operations.

Move forward: The robot moves forward. If the robot moves into an object, it crashes (turns black and ceases to function).
Turn left: The robot turns 90 degrees to the left and doesn't move
Turn right: The robot turns 90 degrees to the right and doesn't move
Check if the front has a WALL or WINDOW: The robot senses the space in front of it. If there is a wall or window directly in front of it, it returns true, otherwise it returns false.
Check if the side has a WALL: The robot senses the space to the RIGHT of the robot. If it senses a wall, it returns true. If there's a window or nothing at all, it returns false.

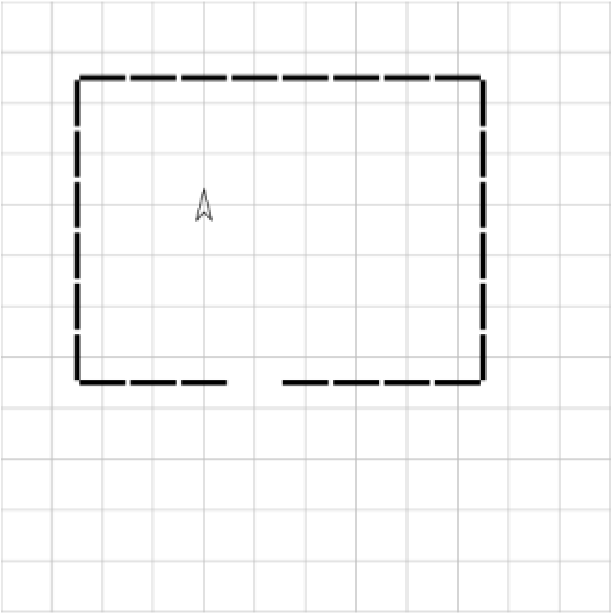
Carol has been placed into a rectangular room with a single window. The width and depth of the room, the position of the window, and Carol's position can be arbitrary. The following program has been designed to position Carol next to the window (the window would be to the right of Carol).

The program is presented in flowchart notation. The rectangles are actions to take. The hexagons denote method calls.



Simulating the Algorithm

Execute this program starting with the following configuration.



Scribe: call out the steps from the flowchart. Begin by calling out the steps from the main method's flowchart since the main method is the starting point of every Java program.
Driver: Move the robot on a sheet of paper, following the instructions given by the scribe.

Both Driver and Scribe: Describe in english how this program works. Give separate descriptions for the main program and for each separate flowchart.
As a team, find a solution to the following: (Note that Carol stops next to the window and cannot go through it.)

1. "Main method"
    a. Starting the main method calls the "go to wall" method, then turns left, then calls the "find window" method, and then the method stops.
2. "Go to wall" method
    a. Starting the method checks whether there is a wall in front of it.
        i. If there ISN'T a wall in front, it will keep moving forward until there is a wall.
        ii. Once there is a wall in front of the robot, the method is finished.
3. Find Window
    a. It checks if there is a wall
        i. If there isn't method returns
    b. If there is, the robot checks if there is a window or wall in front of it
        i. If there isn't, then the robot moves forward and uses the method Find Window again.

c. If there is, then the robot turns Left, moves forward then uses the method Find Window again

Question: In step 1, you should have successfully completed the program. That is, Carol should have found the window and stopped beside it. However, there can be many different starting situations—that is, different positions for the window and different starting locations for Carol. There is at least one starting situation for which the program does not work correctly. Can you find it? If so, what is the problematic starting situation? Is there more than one such problem situation? Try different locations for the window and different starting positions for Carol. Use pencil and paper for sketches.

1. Can you find it? If so, what is the problematic starting situation?
    a. If the window starts at a horizontal position on the bottom-left corner, the robot will keep going around the rectangle.
2. Is there more than one such problem situation?
    a. Yes, there are at least three more problematic situations.
        i. Window vertical position, bottom right corner.
        ii. Window horizontal position, top right corner.
        iii. Window vertical position, top left corner.

After you have found the problem situation(s) in step 4, think about how you did it. If you had to tell someone else how to look for problems like this, what would you say?

Try to think of how your main method works. What does it rely on to function properly? In this case, it would be the walls and windows, as well as the position of carol. So a good idea would be to try and find the most niche situations, like the corners for example.

Suppose you want to change the program, so it works correctly in all situations, including the problem situation(s) you just discovered. How might you do this? There may be more than one possible answer.

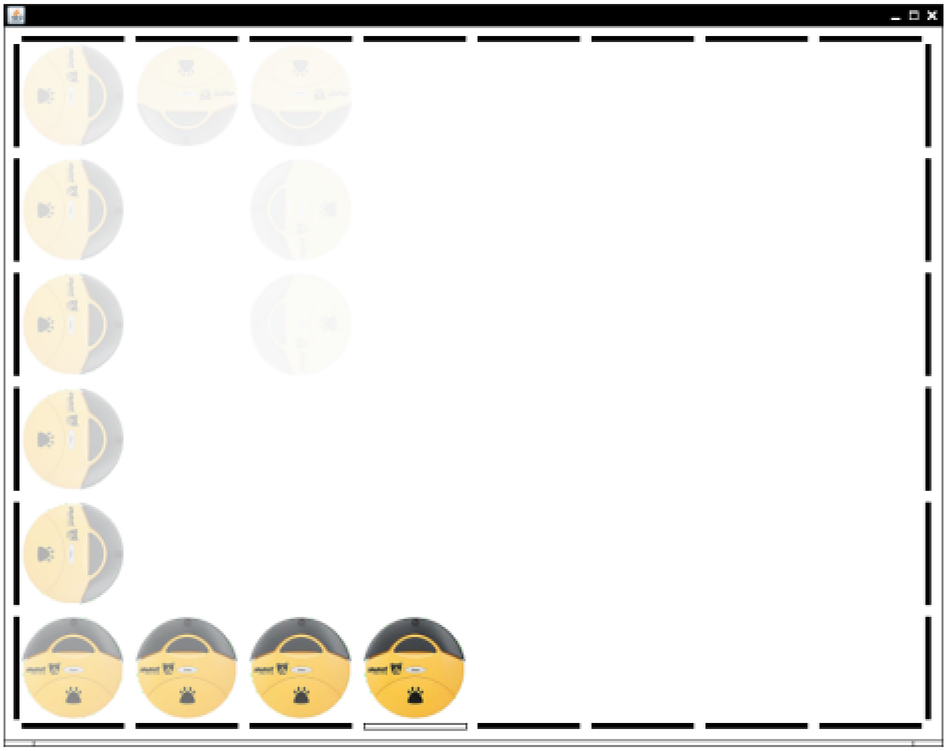
1.
    a. Add another check if right side has window right after the turn left in the find window function

Running the Program

Driver: Open the lab08Carol project, then compile the RobotWallWindowProg class and run the main() method of the class to make sure that it works. If it doesn't, ask the lab instructor for help.

What does the program do?

It did the above Carol functions.



Compile class RobotWallWindowApp1 and run its main() method.

Scribe: What happens when the program runs? Is it one of the situations you identified in Step 3 of Part A? Which method in the CarolWallWindow class should be updated so the robot stops in front of the window?

Yes, it is the case when the horizontal bottom left corner is the window. The findWindow method must be fixed.

Once you identify it, update the method.

Driver: What is your code after the program works correctly?

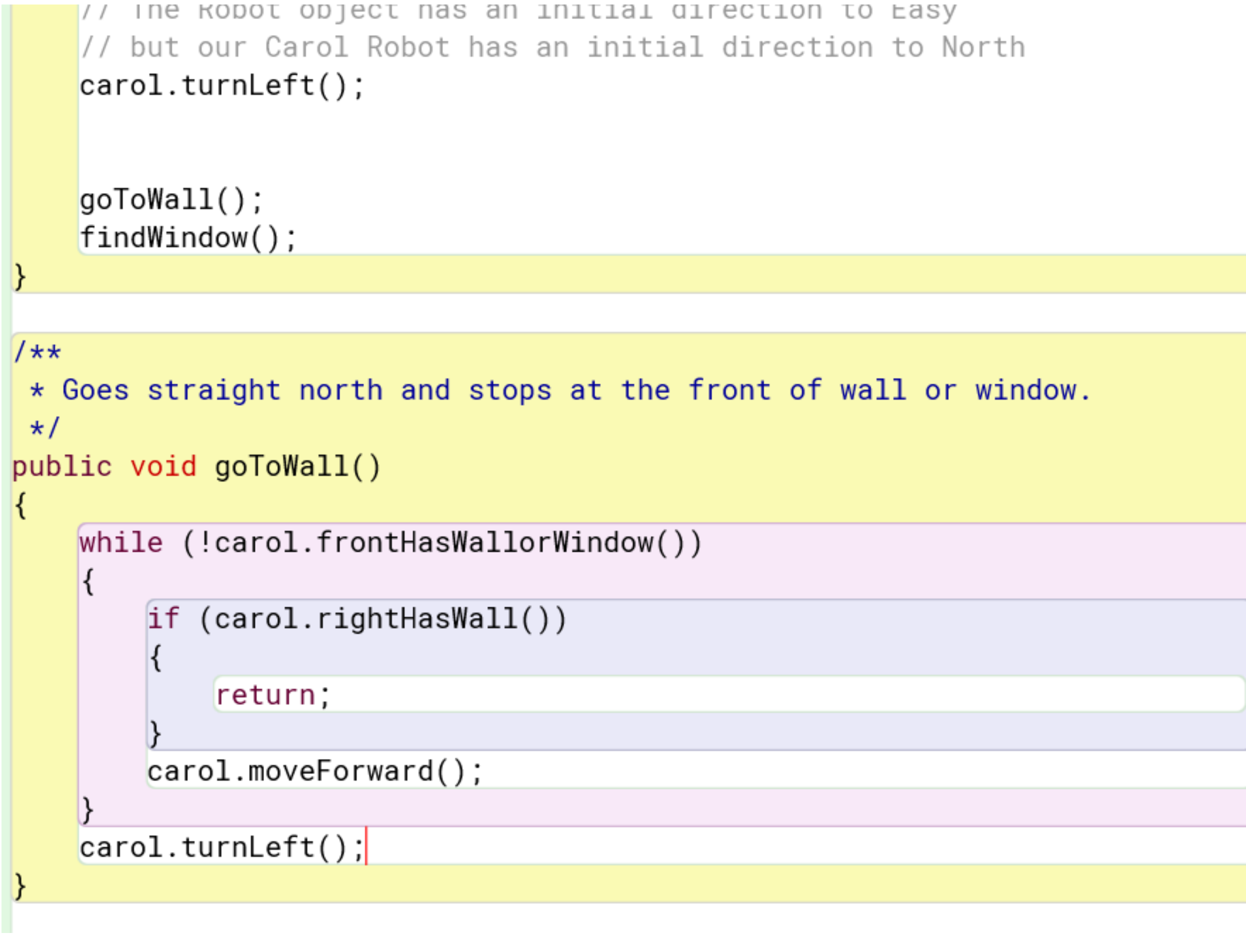
Compile class RobotWallWindowApp2 and run its main() method.

Scribe: What happens when the program runs? Is it one of the situations you identified in Step 3 of Part A? Which method in the CarolWallWindow class should be updated so the robot stops on the first time it moves next to the window?

It will eventually reach the window, just not the first time it moves next to the window. We did not identify it because we thought that so long as it eventually stops next to the window it is successful. We can adjust the goToWall function to make it work.

It is allowed to have the Carol Robot turn right and/or left to decide if the starting position is next to the right or left wall. Once you identify the method that needs fixing, update it.

Driver: What is your code after the program works correctly?



```
    // The Robot object has an initial direction to Easy
    // but our Carol Robot has an initial direction to North
    carol.turnLeft();


    goToWall();
    findWindow();
}

/**
 * Goes straight north and stops at the front of wall or window.
 */
public void goToWall()
{
    while (!carol.frontHasWallorWindow())
    {
        if (carol.rightHasWall())
        {
            return;
        }
        carol.moveForward();
    }
    carol.turnLeft();
}
```

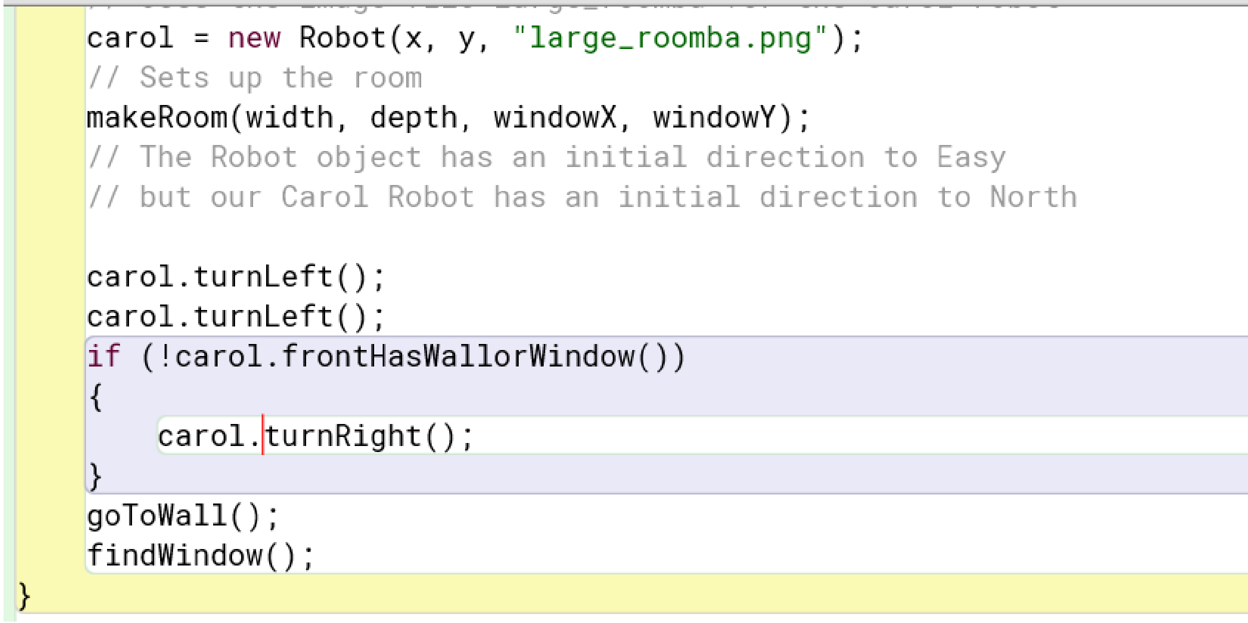
Compile class RobotWallWindowApp3 and run its main() method.

Scribe: What happens when the program runs? Is it one of the situations you identified in Step 3 of Part A? Which method should be updated?

It runs into the wall and turns left twice, but it is not one of the situations we identified. Main method was changed.

The Carol Robot should move counterclockwise along the wall. So it may be necessary to check the starting position and change the direction to south if the starting position is next to the left wall.

Driver: What is your code after the program works correctly?



```
    carol = new Robot(x, y, "large_roomba.png");
    // Sets up the room
    makeRoom(width, depth, windowX, windowY);
    // The Robot object has an initial direction to Easy
    // but our Carol Robot has an initial direction to North

    carol.turnLeft();
    carol.turnLeft();
    if (!carol.frontHasWallorWindow())
    {
        carol.turnRight();
    }
    goToWall();
    findWindow();
}
```

Before you Leave:

Submit the following:

Your lab report with all related screenshots.

Person of the week:

Please state which department you would like to talk to.
Sales
Front Desk
Management

This is a voice line that you may have heard while trying to get in contact with some business. A human was unable to be reached, and you’ve instead met with a robot trying to understand your human speech.

Marian Croak invented a technology named Voice over IP (VoIP) capable of converting voices into digital signals. These tools (while annoying when you’d like to speak to a human) open up a huge number of tools, including skype or whatsapp, and other internet related communication tools.