18-642 Recitation #4

September 21, 2018
Updates

• Homework:
  – Last week homeworks graded on canvas
  – Next week homeworks due Wednesday night
• Homework grading
  – Points are mostly for effort
  – Read comments on canvas, even if you got full points
  – Full points does not mean you got the right answer
    • We’ll try to cover some common issues in recitation
    • If you’re not sure – ask!
Updates

• Projects:
  – Project 3 graded on canvas
  – Project 4 due tonight
    • Remember that code must comply with every Project 3 Checklist item
    • Issues from peer reviews should all be fixed
  – Project 5 released, due in a week
Updates

• Exam #1 in less than a month
  In class on Thursday Oct 11, 2018
Today

• Project 5
• Lightning Round
  – HW #10-3; #11-2a&b
  – (Find your handed in answers to these homeworks NOW if you don’t remember what you said in them)
Project #4 Questions?
Project 5

• Keep track of how many times the turtle has visited a cell
• Split code into turtle and maze components
• Must build and solve m1.maze
• Try running on new maze: m2.maze
  – You do not have to solve this maze
Cell Visit Counts

- You are given a new parameter for displayTurtle: `displayTurtle(int nw_or, int visits);`
- Call this function before you return in your main routine
- Function is for debugging/visualization only – accuracy depends on how well you are keeping track of data locally
student_maze and student_turtle

- student_maze.cpp
  - Does not decide how turtle moves
  - Translates moves from student_turtle into absolute coordinates
  - Think of this like building a simulator for the turtle

- student_turtle.cpp
  - Does not know its absolute coordinates
  - **cannot ask student_maze where it is or absolute orientation**
    - cannot call bump(x1,y1,x2,y2)
    - student_maze should tell student_turtle if a move resulted in a bump
  - Decides how to move based on current state and whether it has bumped
  - This is how a real robot (or human dropped into a maze) would behave
Example with 3x3 Maze

• Turtle is dropped into maze
  – Doesn’t know where or which cardinal direction it’s facing
  – Will draw its own map of the maze (2D array) – like a human carrying graph paper around

Ground Truth
Drawing its own maze

- Turtle doesn’t know where it is: could be anywhere in a 3x3 maze
- Knows it can move at most two squares in any direction from start point because 3x3 maze
- Therefore needs only a 5x5 array to keep track of any moves it makes (assuming it starts in the center of its array)
- Doesn’t know which direction it faces
  - Can assume “local north”

Real maze could be any of these (in rotation) or more!
Translating Actions: Start

Ground Truth
(student_maze)
(x,y,orient) = (1,0,east)

Turtle’s local model
(student_turtle)
(x’,y’,orient’) = (2,2,north)
Translating Actions: turn right

- Student_turtle decides to turn right
- Updates its local orientation \( or' \)
- Tells student_maze it turned right, which updates absolute orientation \( or \)

Ground Truth
\[
(x, y, or) = (1, 0, \text{south})
\]

Turtle’s local model
\[
(x', y', or') = (2, 2, \text{east})
\]
Translating Actions: check bump

- Student_turtle wants to check for bump
- Student_maze sends whether there’s a wall in front of turtle based on x,y,or

\[ \text{Ground Truth} \]
\( (x,y,or) = (1,0,\text{south}) \)

\[ \text{Turtle’s local model} \]
\( (x’,y’,or’) = (2,2,\text{east}) \)

Bump = false

Cool, I can move forward
Translating Actions: Move Straight

- Student_turtle decides to move straight
- Updates its local coordinates \( x', y' \) and local array of visits counts
- Tells student_maze it moved straight, which updates absolute coordinates \( x, y \) and visits count at \( x, y \)
Review:

- student_maze calls decision function in student_turtle
  - Hint: pass return value of bumped() to turtle using this function
- student_turtle decides how to move based on internal logic
- student_turtle updates internal array of cell visit counts
- student_turtle returns an action to student_maze
- student_maze translates to absolute coordinates, calls displayTurtle(), etc
Sequence Diagrams Preview

- You will see this in class this week
- Describes a scenario (interaction between components)
- Time flows top to bottom
- Boxes indicate components
- Arrows indicate messages/calls from one component to another
- Use them to describe interaction between components in Proj 5 writeup

```
<table>
<thead>
<tr>
<th>ece642rtle</th>
<th>student_maze</th>
<th>student_turtle</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>display_turtle(x,y)</td>
<td>action_request(bumped)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>action(turn_dir)</td>
</tr>
</tbody>
</table>
```
Project 5 Questions?

• Review:
  – Keep track of the times a cell has been visited
  – Split into maze and turtle
  – Run on new maze (but don’t have to solve)
    • You’ll need to solve it in a future project

• Builds a foundation for Project 6
  – Will introduce more mazes not solvable by LHR/RHR
Homework Review

- SCC vs MCC
- Equivalence Classes
- MCDC
- State Charts
MCC and SCC

• McCabe’s cyclomatic complexity
  – Counts # of if/while/for conditionals in the code

• Strict cyclomatic complexity
  – Includes +1 for every condition within a branch

• if (a < 0 && b > 0) adds +1 to MCC, +2 to SCC
MCC/SCC in student.cpp

- Check whether your tool computes MCC or SCC
- Find branch statements with multiple conditions
  ```cpp
  mod = true;
  if (z == true && aend == false) {
    if (nw_or == 1) pos_.setY(pos_.y() - 1);
  }
  ```
- Add # of extra statements to MCC to get SCC
  - Or subtract from SCC to get MCC
- For student.cpp, SCC = MCC + 1
Equivalence Classes

• Any input in an equivalence class is expected to cause the code to behave the same
  – If input a and input b are in the same equivalence class, they will both take the same branches in the code
• Always remember special case values:
  – NULL for pointers
  – NaN for floats
  – Overflow values (INT_MIN, INT_MAX)
To achieve MC/DC coverage

if((a > 5) && ((b < 17) || (b > 97) || (b == 42)))

- Inputs to make entire decision true and false
- Inputs to make each conditional true and false
  - a > 5, b < 17, b > 97, b == 42
- Each condition affects decision independently
  - Hold all other conditionals fixed at some value
  - Changing Boolean value of the condition changes the outcome of the decision
  - Ex: b < 17 affects decision?
    - Hold a = 6 (making a > TRUE), b < 42 (making b > 97 FALSE, b == 42 FALSE)
    - If b < 17, decision is TRUE, else decision is FALSE
Questions?
10-3a. Let's say you have a function that takes 25 input parameters. When is it OK to pass those parameters via globals? Why?

10-3b. You have a global array of 100 elements. Should that be counted as 1 global, 100 globals, or something else? Why?

10-3c. You have a struct with 100 different named elements. Should that be counted as 1 global, 100 globals, or something else? Why?

11-2a. What is an argument for including switch statements in a cyclomatic complexity metric?

11-2b. What is an argument for NOT including switch statements in a cyclomatic complexity metric?