Welcome back to CS 5!

Average of these two?

Problem 0: Reading + response...
Problem 1: Four-fours program: Can be done for lab...
Problem 2: Rock-paper-scissors program (Maybe done already!)
Problems 3-4: Picobot! empty room (3) maze (4)

Homework 1
due Mon. night (11:59pm)

The challenge of programming...

syntax
How it looks
semantics
What it does
intent
What it should do

human-typed input
machine-produced output

human-desired output

Another language already?

Python
General-purpose language
you might see 50% by the end of the term even then, <1% of its libraries!

Picobot
Special-purpose language
you'll see 100% in the next 10 minutes

The Picobot simulator
www.cs.hmc.edu/picobot

Surroundings

Picobot can only sense things directly to the N, E, W, and S

For example, here its surroundings are

N x W x E x S

Surroundings are always in NEWS order.
What are these surroundings? Surroundings are always in NEWS order.

N E W S
NxWx

State
I am in state 0. My surroundings are xxWS.

State and surroundings represent everything Picobot knows about the world.

Surroundings
How many distinct surroundings are there?

$2^4 = 16$ possible

Rules
Picobot acts through a set of rules. Each rule expresses your intent for Picobot!

If Picobot’s in state 0 seeing xxWS, then move North, and "change" to state 0.
Wildcards

Asterisks * are wild cards. They match walls or empty space:

I only care about NORTH being EMPTY

N must be empty

EWS may be wall or empty space

direction

new state

current state

surroundings

0 x*** → N 0

8 surroundings in one rule

that's it
1. Run Picobot! Which rule A, B, or C runs first? _____
   1a. How many times does rule (A) run? ______
   1b. How many times does rule (B) run? ______
   1c. How many times does rule (C) run? ______

2. Picobot stops when no rule matches. **Where does it stop?**

3. Add a rule so that Picobot continues **back upward!**

---

**The Rule is One step per rule**

<table>
<thead>
<tr>
<th>state</th>
<th>surr.</th>
<th>move</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 N***</td>
<td>W</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0 x***</td>
<td>N</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1 ***x</td>
<td>S</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

Extra #1 | Rule A has a bug! What is it?
Extra #2 | Add rules to finish exploring the empty room from any starting point...
Extra #3 | How to do this in only 6 rules total?!
Warning!  *What's wrong here?*

<table>
<thead>
<tr>
<th>state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x***</td>
<td>S</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>***x</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

Notes: Picobot checks its rules from the top each time. *When it finds a matching rule*, that rule runs.

**Maze strategies?**

CS ~ *Complexity Science*

Problem 3: Shortest Picobot program:

6 rules

Problem 4: Shortest Picobot program:

8 rules

Suppose Picobot wants to traverse a maze with its right hand always on the wall...

(A) CORRIDOR rule

*If you’re facing N with a wall at right and space ahead then go forward*:

<table>
<thead>
<tr>
<th>state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xE**</td>
<td>N</td>
<td>0</td>
</tr>
</tbody>
</table>

(B) INTERSECTION rule

"If you’re facing North and lose the wall then get over to the wall now!"

<table>
<thead>
<tr>
<th>state</th>
<th>surroundings</th>
<th>direction</th>
<th>new state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

(C) DEAD END rule

Write 1 or 2 rules to tell Picobot to do the right thing if it hits a dead end.

Repeat this IDEA for all four states, representing all four facing directions.
(1) Find and correct as many errors as you can in this code:

```python
import random

user = input("Choose your weapon! ")
comp = random.choice(["rock", "paper", "scissors"])  
print('user (you) chose:', 'user')
print('comp (me!) chose: ' comp)

if user == rock and comp = 'paper'
    print('The result is, YOU LOSE.'
     print('unless you're a CS 5 grader, then YOU WIN!')
```

(2) This one line does **three** things... what are they?

(3) Extra! Can you find 7 punctuation marks used in *more than one way* here?