355 fall 2022 assignment 2

For this assignment, post CLARIFYING QUESTIONS ONLY on eclass. Posting suggestions or your answers or hints on eclass or any other site – e.g. a discord server or anywhere else – is plagiarism.

You can work on this assignment in groups of up to 5: within your group, you can discuss any questions, but you cannot copy answers. Each student must submit their own assignment. Discussing or copying with any student outside your group is plagiarism.

We might ask you later to explain your answers: if you are unable to do so, we might deduct some or all marks and report this to the faculty of science.

For this assigment, each student's secret number is the 4th and 5th integer of their student id, interpreted as a 2-digit number. E.g. if your id is `***91**`, then your secret number is 91. Some questions ask you for m, your secret number mod 3. E.g. if your secret number is 91, your m is 1.

Submit each answer on eclass. There are 7 questions, each worth 1 mark.

1. **If you do not answer this question we will not mark the assignment and your assignment score will be 0.**

   (a) In your own words, state that you accept the plagiarism policy above.

   (b) Give the names and ccids of all members of your discussion group (including yourself). Explain how you worked together: e.g. discussed every question, discussed only questions 1 and 3 with group members X and Z, etc.

2. For each 2×2 game below, is it legal? If yes, write yes. If no, give the first illegal move, and explain why the move is illegal (e.g. superko violation? no liberties? pass violation?).

   if your m is 0, use these games:

   game A: 1.B[pass] 2.W[pass]

   game B: 1.B[pass] 2.W[b2] 3.B[a2] 4.W.[a1] 5.B[pass] 6.W.[b1] 7.B[pass] 8.W[pass]

   game C: 1.B[a2] 2.W[b2] 3.B[a1] 4.W[b1] 5.B[a2] 6.W[a1] 7.B[a2] 8.W[b1] 9.B[pass] 10.W[pass]

   if your m is 1, use these games:

   game A: 1.B[pass] 2.W[pass]

   game B: 1.B[pass] 2.W[a2] 3.B[b2] 4.W.[b1] 5.B[pbss] 6.W.[a1] 7.B[pbss] 8.W[pbss]

   game C: 1.B[b2] 2.W[a2] 3.B[b1] 4.W[a1] 5.B[b2] 6.W[b1] 7.B[b2] 8.W[a1] 9.B[pass] 10.W[pass]

   if your m is 2, use these games:

   game A: 1.B[pass] 2.W[pass]

   game B: 1.B[pass] 2.W[b2] 3.B[a1] 4.W.[a2] 5.B[pass] 6.W.[b2] 7.B[pass] 8.W[pass]

   game C: 1.B[a1] 2.W[b2] 3.B[a2] 4.W[b2] 5.B[a1] 6.W[a2] 7.B[a1] 8.W[b2] 9.B[pass] 10.W[pass]

3. Python3 go program `gp.py` in the class github repo represents each go position using a guarded board. A simpler way to represent go positions is with unguarded boards. E.g. below left is an empty 3x4 go goard, below right are the locations of string of length 12 that represents it: point a1 is stored at location 0, point b1 is stored at location 1, point c1 is stored at location 2, etc.

```
3  .  .  .  .       8  9 10 11
2  .  .  .  .       4  5  6  7
1  .  .  .  .       0  1  2  3
   a  b  c  d
```

   Assume that we change `gp.py` to `gpsimple.py` by using unguarded boards instead of guarded boards. Rewrite function `empty_board(r, c)` as it should appear in `gpsimple.py`. Explain the changes that you make.

4. Continuing from the previous question, give the new version of the following code fragment from `tromp_taylor_score` as it should appear in `gpsimple.py` in the class github repo. Explain the changes that you make.

```
for j in self.nbr_offsets:
            x = j+q
            b_nbr |= (self.brd[x] == BLACK)
            w_nbr |= (self.brd[x] == WHITE)
            if self.brd[x] == EMPTY and x not in empty_seen:
              empty_seen.add(x)
              empty_points.append(x)
              territory += 1
```

5. a) I created the function `m.py` in directory `go` of the class github repo by making some changes to `maze.py`. In your own words, explain what changes I made. Hint: the linux function `diff` is your friend.

   b) `m.py` is a randomized algorithm: comment out one line to make it deterministic. Explain which line you commented out and why. Use this deterministic program in the rest of this question.

   If your m is respectively 0, 1, 2, answer question c), d), e) only.

   c) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in clockwise order, starting with the right neighbor. Give the changed line and then show the output when you execute the program with input `m90.txt` .

   d) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in clockwise order, starting with the bottom neighbor. Give the changed line and then show the output when you execute the program with input `m91.txt` .

   e) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in counter-clockwise order, starting with the top neighbor. Give the changed line and then show the output when you execute the program with input `m92.txt` .

6. Consider the recursive maze traversal function `rwander( )` in program `rmaze.py` . If your m is respectively 0, 1, 2, answer question a), b), c) only.

   a) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in order left, top, bottom, right. Counting from 0, label the empty cells in `m90.txt` in the order in which they are searched by `rwander( )`.

   b) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in order top, bottom, right, left. Counting from 0, label the empty cells in `m91.txt` in the order in which they are searched by `rwander( )`.

   c) Modify line `nbr_offsets = [(0,-1), (0,1), (-1,0), (1,0)]` so that the neighbors of each cell are considered in order bottom, right, left, top. Counting from 0, label the empty cells in `m92.txt` in the order in which they are searched by `rwander( )`.

   E.g. for a), your answer might look something like this:

```
X  X  X  X  X  X
X  4  1  6  9  X
X  0  +  2  7  X
X  5  3  8 10  X
X  X  X  X  X  X
```

7. If your m is respectively 0, 1, 2, answer question a), b), c) only. Consider the life simulation `life.py` in directory `life`. This program using a floating coordinate system: it can change with every tick. Assume that the coordinate system is fixed. Here is your starting position at tick 0:

```
a)                      b)                      c)
    -2-1 0 1 2              -2-1 0 1 2              -2-1 0 1 2
 2   . . . . .          2    . . . . .          2    . . . . .
 1   . . * . . 1        1    . * * * .          1    . * * * .
 0   . * . . . 0        0    . * . . .          0    . . . * .
-1   . * * * .-1        1    . . * . .         -1    . . * . .
-2   . . . . .-2        2    . . . . .         -2    . . . . .
```

   a) At tick 1000, show the 3×3 subgrid containing all live cells.

   b) At tick 1001, show the 3×3 subgrid containing all live cells.

   c) At tick 1002, show the 3×3 subgrid containing all live cells.

   Label the rows and columns of your subgrid correctly, so like this (but with different labels and a different pattern):

```
     -555 -554 -553
  72   .    *    .
  73   *    *    *
  74   .    *    .
```

   Running `life.py` on inputs `t/g0`,`t/g1`,`t/g2`, might help you with this question.