Homework 10 - Problem Description & Literature Review

Andrew Carter and Beryl Egerter

April 9, 2013

1 Problem Descriptions

1.1 Q1

1.2 Q4

1.2.1 Summary

Interview Question 4 (Q4) asks the interviewee to locate and fix a bug in a program that they are told plays connect four. The bug is that when pieces are dropped into the board, the program lets pieces be dropped in even if the column is full. The program is 70 lines long and covers two pages when printed.

1.2.2 Copy of Problem

The problem can be found in the Appendix attached below as Interview Question 4.

1.2.3 Solution Description

To find the correct solution we start at line 49 where the code starts executing. Some variables are set up for use within the Read-Eval-Print loop that follows. Here the Board is initialized. It calls the __init__ function (line 3) which create a list of lists stored as the self.board variable. The initial lists are empty. It also stores the height and width of the board in class variables.

At line 55 we see that the program asks for input, the read part of the loop. Then, it calls board.drop(player,c). In the drop function (line 8), we can see that if the column selected is part of the board, the piece is appended to that column in the list of lists. This is analogous to dropping a piece into the board, meaning this is where the bug needs to be fixed by.

Since we check that the column is valid, we can also check that the column is not full by changing the drop function to include this line between lines 9 and 10: if len(self.board[column]) < self.height:, to make sure that pieces will not be dropped.

Other possible solutions that cause the drop function to not be called if the column is full would also be acceptable.

2 Literature Review

3 Appendix (for reference)

3.1 Interview Question 1

Code given to participants did not include line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale.
Verbal prompt was given before handing code to participant:
"For this question we would like you to familiarize yourself with some Python code. Please explain to us what you think this code does."

```
0   def func2(list , num):
1     return func1(list , num, func4)
2
3   def func4(a, b):
4     return a * b
5
6   def func1(list , num, f):
7     acc = 0
8     for i in list:
9         acc += f(i, num)
10    return acc
11
12  def main():
13    print(func3([1,2,3,4]))
14
15  def func3(list):
16    return func2(list , 4)
17
18  main()
```

3.2 Interview Question 4

Code given to participant did not contain line numbers.
Code given to ACBE1, ACBE2, and ACBE3 was in color. Code given to ACBE4 was in grayscale. Code given to ACBE1 and ACBE2 had double equals signs that were joined together. Code given to ACBE3 and ACBE4 had spaces between the equals signs.
Verbal prompt was given before handing code to participant:
"For this question we would like to have you look at some code in Python. This is the scenario: You acquired a connect 4 program from a friend. However, the friend has warned you that you can put too many pieces in a column. Determine a possible fix for this bug so that you can enjoy your connect 4 program."

```
0   #!/bin/env python3
1
2   class Board(object):
3     def __init__(self, width=7, height=6):
4       self.board = [[] for i in range(width)]
5       self.width = 7
6       self.height= 6
7
8     def drop(self, player, column):
9       if column < len(self.board):
10        self.board[column].append(player)
11        return True
12      return False
13
14    def __str__(self):
15      result = ""
16      for r in reversed(range(self.height)):
17        result += "|"
18        for c in range(self.width):
19          if r < len(self.board[c]):
20            result += self.board[c][r]
21          else:
22            result += "␣"
23          result += "|"
24        result += "\n"
25      result += "-" * (2 * self.width + 1)
26      return result
27
28    def full(self):
29      return all(len(col) >= self.height for col in self.board)
30
31    def score(self, player):
32      for c in range(self.height):
33        for r in range(len(self.board[c])):
34          p = self.board[c][r]
35          for dc,dr in ((0,1),(1,0),(1,1),(1,-1)):
36            for i in range(1,4):
37              nc = c + i*dc
38              nr = c + i*dr
39              if nc < 0 or self.width <= nc:
40                break
41              if nr < 0 or len(self.board[nc]) <= nr:
42                break
```

```
43              if self.board[nc][nr] != p:
44                break
45            else:
46              return 1 if p == player else -1
47      return 0
48
49  other = {'X' : 'O', 'O' : 'X'}
50  player = 'X'
51  board = Board()
52
53  while True:
54    try:
55      c = int(input("%s -> " % player))
56    except TypeError:
57      continue
58    if not board.drop(player,c):
59      continue
60    print(board)
61    if board.score(player):
62      print("Player %s Wins!!!" % player)
63    elif board.full():
64      print("Tie")
65    else:
66      player = other[player]
67      continue
68    board = Board()
69    player = 'X'
70    print(board)
```