Final exam

due Mon May 8 at midnight, by email

Instructions

This is a **take-home exam.** You may use whatever materials you like, including the source code that we have written and any notes/links provided. If you use quotes or ideas from any source other than your own mind, you **must cite the source** and use quotation marks appropriately.

You may **not** do this exam in groups. Your responses must be entirely your own. If I notice suspicious similarity between the answers of two or more students, or if your answers are copied from another source, your score will suffer.

All of your answers should be **concrete** and pertain **specifically** to the project we implemented this semester. I expect about a page of explanation per question. Submit your answers by email to `christopher.league@liu.edu`.

Questions

1. Our Minesweeper game consists primarily of two classes:

- `Constraints`, which tracks the constraints on sums of Boolean variables in order to assist the AI player with making moves; and
- `Minefield`, which tracks the locations of mines, spaces that are cleared, and results shown to the player. It also determines when the game is over. In short, it implements the rules of the game.

The interactions between the game and the human or AI player are mixed in to the `Minefield` class and the `__main__` chunk of `sweeper.py`.

One additional feature that our code could support is to have a graphical user interface (GUI), on which a human player could click with a mouse or touch with a finger.

Describe in detail how you could incorporate the GUI and redesign the human and AI player code to promote **cohesion** and reduce **coupling** between all these components. Consider using diagrams if it helps elucidate the connections between components.

2. I used some ‘doc’ tests in the `Constraints` class to illustrate and verify some of the simplifications I expect it to do. For example, this doc test:

```
>>> cs = Constraints()
>>> cs.add(3, "abc")
>>> cs.show()
a=c=b=1
```

confirms that given $\sum\{a, b, c\} = 3$ we can conclude that $a = 1$, $b = 1$, and $c = 1$.

What tools and techniques would you use to ensure that we are exposing most of the possible sources of bugs in this code?