(1) What is the problem you have formulated in Part B? If this is not a well-known problem, explain what the problem is and any rules you have made up.

Instant Insanity

(2) What is your state representation and why?

A list of cubes that have lists of color of each face.

Yaw clockwise:0 to 4 to 2 to 5

Pitch clockwise:0 to 1 to 2 to 3

Roll clockwise: 4 to 1 to 5 to 3

Pitch

\_4\_

| 0 | 1(Yaw) | 2 | 3 Roll

\_5\_

Face 3 is the Bottom, Face 1 is the Top

Each face is given an index from 0 to 6.

Color are represented by numbers from 0 to 3

0:Red 1:Green 2:Blue 3:White

(3) What operators do you provide? Did you face a choice when you designed your operators, and if so, what is another possibility that you considered? Why did you go with the choice you made?

0: Rotate First cube for 90 degree Clockwise in Yaw direction

1: Rotate First cube for 90 degree Forward in Pitch direction

2: Rotate First cube for 90 degree Clockwise in Roll direction

3: Rotate Second cube for 90 degree Clockwise in Yaw direction

4: Rotate Second cube for 90 degree Forward in Pitch direction

5: Rotate Second cube for 90 degree Clockwise in Roll direction

6: Rotate Third cube for 90 degree Clockwise in Yaw direction

7: Rotate Third cube for 90 degree Forward in Pitch direction

8: Rotate Third cube for 90 degree Clockwise in Roll direction

9: Rotate Fourth cube for 90 degree Clockwise in Yaw direction

10: Rotate Fourth cube for 90 degree Forward in Pitch direction

11: Rotate Fourth cube for 90 degree Clockwise in Roll direction

Initially, we tried to use 24 operators, clockwise and anticlockwise 90 degrees’ rotation in three directions of four cubes. However, it will be of great annoyance to user to choose from 24 options. Since rotating 90 degrees clockwise for three times equals rotating 90 degrees anticlockwise, we decided to use 12 operators for simplicity: only clockwise 90 degrees’ rotation in three directions of four cubes.

(Exception: forward rotation is used in pitch rotation, since pitch rotation has no clockwise or counterclockwise rotation)

(4) Did you provide a visualization in your formulation? Briefly, how does that work?

Yes. We use Missionaries\_Array\_VIS\_FOR\_TK.py as our template to use functions and classes in show\_state\_array.py. The window has an array, on which flat patterns of four cubes is displayed. The sides of tower stacked by the four cubes are displayed on the right side of array.

(5) Describe how you and your partner(s) divided up the work of the project. Make sure that both partners' full names are included in the report. (Note that both of your names should also be included as authors, within the METADATA portion of the problem-formulation template file.)

We first discussed about which problem to choose and outline of the code together, then broke down the task and assigned them. Ethan Jiang mainly focus on formulation of the problem, coding class cube; Marco Xu mainly focus on coding Operators, methods, and class state; Alan Li mainly focus on visualization and debugging.

(6) Describe any particular challenges you had with this project.

Flaws in show\_state\_array.py:

As we were trying to create my visualization files for FarmerFox and Instant Insanity in P2 with show\_state\_array.py provided with P2, the visualization of FarmerFox and Missionaries and Cannibals worked well, whereas the program became slower and slower as we made more steps in visual InstantInsanity game that uses a bigger array.

It turned out to be the problem of show\_state\_array.py. The original program creates new state\_array every time it renders new states and uses create\_rectangle() and create\_text() to display the array. The create\_rectangle() and create\_text() functions create new items that would cover previously created items. However, items created previously won’t be deleted and will still consume computing power and memory each time the window updates, therefore slowing down the entire program.

We eventually solved the problem by changing the behavior of state\_array class from creating new items each time it updates window, to changing rectangle items created during initialization.

While we were doing debugging, there were 3 major bugs. The first was the yawRotation method which is supposed to rotate the cube clockwise doing rotation counterclockwise. Later on we found out that we made the wrong order with the temp variable assignment. The second one was that in the GUI coding, we found out that the ‘sides’ variable in the window was not showing the right cube’s color. We realized that it is because in the double loop, the iterating var was actually representing the first 4 faces of the cube, not the 4 faces that we wanted to display. The last bug is that the ‘back’ command is not working. This is because we did not define ‘old’ state, so every time STACK tries to retrieve the old state it got nothing.