MATH482 SPRING 2019

NAME(S) ____

Programming Project-minimal obstacles in Instant Insanity

OUT: Tuesday, March 12, 2019

RET: Due Thursday, May 9, last session (30/30)

Design a computer program to search for "obstacles," if they exist, for Instant Insanity puzzles of size **30** cubes. **If an obstacle exists find the smallest one.** As mentioned in class, by obstacle I refer to a subset of the cubes, which cannot be part of any solution. I.e., any stacking of these cubes entails at least one side having some color showing up two or more times. For instance if a given puzzle has two monochromatic cubes of the same color then that puzzle has an obstacle of size **two**. You have free reign over what type of algorithm to use, but I would prefer for you to write the program in Python as it will be easier for me to understand. Please turn in your source code along with your results. A brute-force algorithm (depth or breadth first search through the tree of possible selections) may work fine for puzzles without solutions of size less than 10 cubes. But in getting to size 30 the combinatorial explosion of possibilities may require some special considerations. Particularly insightful and efficient algorithm design may qualify for extra credit. Note that observing that a particular color only shows up three times is not the same as identifying the smallest obstacle. For this project we are looking for obstacles, not solving puzzles that have solutions. Of course it is possible (although unlikely) that one of the randomly built puzzles below actually *has* a solution. In that case, simply provide the solution.

As promised I am now supplying you with the data for the four puzzles that I will grade.

The routine I describe will fill in the colors in sequence: (**cube1** opposite pair 1 left, cube1 opposite pair 1 right, cube1 opposite pair 2 left, cube1 opposite pair 2 right, cube1 opposite pair 3 left, cube1 opposite pair 3 right, **cube2** opposite pair 1 left, cube2 opposite pair 1 right, cube2 opposite pair 2 left, … , **cube30** opposite pair 3 left, cube30 opposite pair 3 right).

The color-generating assignment formulae are:

$1 + ((\text{floor } n\pi) \bmod 30)$, $1 \le n \le 180$, for puzzle one,

$1 + ((\text{floor } ne) \bmod 30)$, $1 \le n \le 180$, for puzzle two,

$1 + \left(\left(\text{floor } n\sqrt{3}\right) \bmod 30\right)$, $1 \le n \le 180$, for puzzle three,

$1 + \left(\left(\text{floor } n\sqrt{5}\right) \bmod 30\right)$, $1 \le n \le 180$, for puzzle four.

These formulas specify, exactly, which colors land on which faces, so everyone will have the **exact same** data set.

Please provide an English description of your algorithm (pseudocode) for determining the smallest obstacle. Please also provide your source code, and output. The key is an obstacle of minimal size. Be sure to include this in your report. If you cannot solve the problem entirely, then you can give an upper bound on the size of the smallest obstacle, for partial credit. For instance, if you determine that the puzzle does not have a solution, then you can surely say the obstacle size is ≤ 30. But of course I would like to see, for a puzzle without solution, what the **smallest** obstacle is.

As I said earlier, extra credit will be awarded for a particularly clever or insightful algorithm (something beyond brute-force). Also you might compare your input with other groups, just to make sure you are dealing with the same puzzles. If your input is wrong I cannot give full credit!

Finally, I allow you to work in groups. Maximum group size is SIX. Each group will submit one report; each member of a given group will receive the same score on the program. It is probably obvious with no need to say it, but when forming your group you should take into consideration the attendance of the candidate members. If someone has skipped a lot of lectures then they will most likely skip group meetings, too.