COT 3100 Program #1 Spring 2017
Assigned: 3/28/2017
Check WebCourses for due date

Note: This program is purely optional. 5% of the course grade is determined from either attendance in recitation OR two programs. The maximum of the two options will be what each student receives for their grade.

General Program Directions (to be followed for both programs)

Turn in a single source file, in Python, C or Java, with the name designated in the note at the bottom of this page that solves the problem described below. Please read all of your input from standard in and output to standard out. To test your program with input files, please pipe the input file into your program and pipe the output to another file. If you don't know how to do this, please see a TA to describe this process to you.

General Grading Criteria

Your submissions for all programs will be graded based on points in these two categories:

1) Execution Points
2) Coding Style Points

The first criteria will 80% of the program grade. It will simply be based on the number of test cases your program produces correct output on within the designated time limit. (No test cases after a crash will be run to see if your program would have passed them.)

The second criteria will typically be 20% of the program grade. It will look very similar to my Computer Science I style criteria. Use good variable names, indent properly, use reasonable white space, comment appropriately, as well as good coding style (use of methods/functions when appropriate, reasonable problem breakdown and so forth.)

Deliverables

A single source file, named ***tromino.py***, ***tromino.c***, ***tromino.java*** that solves the program stated on the following page, using the input and output formats stated on the following page, using standard input and standard output. The file should be submitted via WebCourses by the due date and time stated in WebCourses.

The Problem: Tromino Tiling

Write a program that tiles a 2^n x 2^n grid with one missing unit square with L shaped tiles (as shown in the induction problem in lecture.) Notice that the tiling can be done in many different ways for most input cases, but to make grading easier, restrictions will be given that "force" one correct tiling. Please carefully follow these directions that follow:

You will be asked to tile squares of a size no bigger than 64 x 64. Each L shaped tile will be designated with a capital letter, so there are only 26 possible tiles that can be printed. Thus for any case larger than 8 x 8, the same tile will be printed multiple times. The first tile placed will be 'A', the second 'B', etc. after tiling with 'Z', go back to tiling with 'A'.

Please write your solution recursively, as indicated by the inductive proof in class.

Here is the order in which you will always tile your cases within your recursive code:

1) First place the "center" tile. Recall that you decide where to place it based on the quadrant that contains the empty hole. The center tile goes in the center squares of the OTHER THREE quadrants.

2) Make 4 recursive calls, first to the top left, then top right, then bottom left, finishing with the recursive call to the bottom right. ***You must make these recursive calls in this order to replicate the correct result.***

In terms of implementation, use a "global" variable that keeps track of the current letter for the tile. Immediately after setting a tile, update this global variable so that the change is reflected for future recursive calls.

Input Format

The first line of the input file will contain a single positive integer, *n*, representing the number of input cases to process. The input cases follow, one per line. Each input case will contain three space separated integers: n ($1 \le n \le 6$), r ($1 \le r \le 2^n$), and c ($1 \le c \le 2^n$), representing that the grid to be filled is of size 2^n x 2^n with a hole at row r, column c, where the top left square is row 1, column 1.

Output Format

For each input case, provide the desired tiling, ***leaving a space character*** in the location of the hole in the grid. Print no extra characters. Follow the output of each grid with a blank line.

Sample Input

```
2
1 2 1
2 3 2
```

Sample Output

```
AA
 A

BBCC
BAAC
D AE
DDEE
```