Middle East Technical University
Department of Computer Engineering

CENG 462
INTRODUCTION TO ARTIFICIAL INTELLIGENCE
Fall 2015
Homework 1

HYPER SQUARE

This puzzle is like the planar version of rubic cube. However, the rules are different. Here is the square you need to deal with (initial configuration) :

1	2	3
4	5	6
7	8	9

The actions that can be taken to alter the square and the resulting configuration for the square is:

- **R11** : rotate 90 degrees C(lock)W(ise) the upper left sub-square

4	1	3
5	2	6
7	8	9

- **R12** : rotate 90 degrees CW the upper right sub-square

1	5	2
4	6	3
7	8	9

- **R21** : rotate 90 degrees CW the lower left sub-square

1	2	3
7	4	6
8	5	9

- **R22** : rotate 90 degrees CW the lower right sub-square

1	2	3
4	8	5
7	9	6

- **C01** : Exchange the row0 & row1

4	5	6
1	2	3
7	8	9

- **C12** : Exchange the row1 & row2

1	2	3
7	8	9
4	5	6

The machine gives you the resulting configuration of the puzzle after some number of actions applied to the first initial configuration i.e. ({1,2,3}, {4,5,6}, {7,8,9}). Hence, your program should take a configuration as input and find the minimum number of steps to the start from initial configuration to reach the given configuration. You have to use **A* search** algorithm with two heuristics; *Manhattan distance* and a heuristic you generated. Your max 2-page report should contain details of your heuristic and the proof showing that it is admissible.

Example input*;*

1 7 9
2 8 3
4 5 6

The output should be; C12 C01 R11. That is starting from the initial configuration we first need to *Exchange the first and second row*, then *Exchange the zeroth and first row* then *Rotate the upper left square* to get the given configuration.

INPUT & OUTPUT SPECS

The first line of the input file gives you the number of **independent** cases to be solved. In the example below, the first line is 2 so there will be cases below. In the beginning of each case there would be an indicator of which heuristic you should use, (M)anhattan or (Y)ours and the following 3 lines will contain the square.
In the output file you need to give the solution i.e. series of actions (to reach the given matrices in the input file from initial configuration) for each of the test cases. *(Remember: There will be one '\n' character after the last number in the hw1.inp, and you should print just one '\n' character after the last solution)*

hw1.inp
2
M
1 7 9
2 8 3
4 5 6
Y
7 8 9

```
4 1 2
6 5 3
```

hw1.out

```
C12 C01 R11
C12 C01 R22 R21
```

Also print the branches generated by your search to *stdout*.

Your code should be in one of the following: C/C++, Python, C#.

Upload your code hw1.cpp/py/cs and report hw1.pdf zipped in a hw1.rar file to COW. You *may* be called to present your homeworks to me which will be arranged later, explaining your algorithm showing your flow of program.