### Question 1

Players X and Y decide to play a game of *K*-in-a-row. Each player chooses piece of either colour black (denoted by B) or white (denoted by W). Turn by turn each player drops her piece in an *N* x *N* table. The pieces drop down to the bottom-most empty slot. Each state in the game can be represented by a 2D matrix containing a ‘B’ where a black piece is, ‘W’ where a white piece is and a dot ‘.’ to indicate the empty slot. A player wins if she can place *K* pieces of her colour in a row, either horizontally, vertically or diagonally. The four possible orientations are as follows for the case of 3-in-a-row:

|  |  |  |  |
| --- | --- | --- | --- |
| W W W | WWW | W  W  W | W  W W |

Now for a given state of the game, you are allowed to rotate the table by 90 degrees in clockwise direction. Once you rotate the table, due to gravity the pieces will fall down to bottom-most empty position. For example,

|  |  |  |
| --- | --- | --- |
| **Initial State** | **After Rotate** | **After Gravity** |
| B.... | BBWBB | BB... |
| BWB.. | WBBW. | WB... |
| WBW.. | WWWB. | WWWB. |
| BBWWW | BW... | BWBW. |
| BWWBB | BW... | BWWBB |

|  |  |
| --- | --- |
|  |  |

For a given state, you have to determine which colour wins after the rotation and gravity effect. The possible results are – “Black”, “White”, “Both” or “None”.

**Program Specifications:**

* You will be given *N,* where 3 <= *N* <= 50 which is size of the table, followed by row size *K*, where 3 <= *K* <= N.
* You will be given a state of the game in a 2D matrix as mentioned above.
* The program should return the result after rotation and gravity effect from the given state. The possible results are as mentioned above.

**Sample inputs:**

1. 3 3  
   B..  
   WB.  
   WB.
2. 4 4  
   W...  
   BW..  
   BW..  
   BW..

1. 7 3  
   .......  
   .......  
   .......  
   ...W...  
   ...BB..  
   ..BWB..  
   .WWBW..
2. 6 4  
   ......  
   ......  
   .W...W  
   .W..BB  
   .W.WBW  
   WB.BBB

**Sample outputs:**

1. Black
2. White
3. None
4. Both

### Question 2

House and team have a critical case at hand. The patient is a U.S. Air Marshall who has lost his vocal skills but the team has a few questions which need to be answered by the Marshall. An easy way for the Marshall to communicate is via Morse code, his experience will come handy. But the problem is, House and team are not so quick to decode what the Marshall is signaling. Here is where you come into the picture. You have to write a program which takes an array of characters as input and outputs the decoded message.

How Morse code works [1]:

* Each letter is encoded as a combination of 0’s, 1’s and/or 2’s.

|  |  |
| --- | --- |
| A | 102 |
| B | 2010101 |
| R | 10201 |
| E | 1 |
| ….. |  |

* Letters in a word are separated by three 0’s.
* Words are separated by seven 0’s.

Program specifications:

* You will be given a character array as input which you will have to decode
* For simplistic purposes, the character set will be restricted only to alphabets (case insensitive)
* The program template given to you will have a mapping of the list of characters and their respective codes.
* The input may contain errors, your program will have to print the decoded message till the error is found and then notify that an error is found by printing ‘-1’.
* For extra karma points, you can improve the speed of the program by constructing a binary tree of 1’s and 2’s and searching the literal found through the tree.

Sample test cases:

Input: 10001010102000100010201000201020200020101010002020200020101000201020200000001020101000101000100010101

Output: EVERYBODY LIES

Input: 10001010102000100010201000201020200020101010002020200020101000201020200000001020101000112

Output: EVERYBODY L-1

[1]: We have modified the actual Morse code rules for simplicity. These are the actual rules:

Each character (letter or numeral) is represented by a unique sequence of dots (1’s) and dashes (2’s). The duration of a dash is three times the duration of a dot. Each dot or dash is followed by a short silence (0), equal to the dot duration. The letters of a word are separated by a space equal to three dots (one dash), and the words are separated by a space equal to seven dots. The dot duration is the basic unit of time measurement in code transmission.