Introduction

For thousands of years, human cultures around the world have practiced various forms of *divination*. Divination refers to attempts to gain insights into questions or situations using a ritualised process. For example, Ifá is a Yoruba religion and system of divination practiced in Nigeria, while Ramal Shastra is a form of Indian Astrology involving specifically designed dice.

One particular practice is *geomancy*, which derives from the Greek word *Gaia* (earth) and *manteia* (prophecy). This involves using elements of the earth, such as rocks or sand, to generate answers to questions. While there is no clear agreement on the origin of the practice, it seems that the general consensus is that it originated in either North Africa or Arabia, and spread through the spice routes to reach West Africa and Turkey. Whatever the origin, the practice and ideas eventually made its way to Europe betwen the years 1100–1200 at the beginning of the Renaissance, where it was classified as one of the *forbidden arts* (along with practices such as necromancy and pyromancy).

In this assignment, we will be replicating the geomancy practice of the Bamana. This process is outlined in the paper available on Moodle (it is not necessary to read the paper, but if you are interested, please feel free).[1] Of course, we are computer scientists, and too much sand will get in our computer vents and mess everything up! So instead, we will stick to a digital replication of the Bamana's procedure.

Figure 1: Anakin doesn't like sand because it's coarse and gets everywhere (like into his RAM modules)!

This form of geomancy is relatively standardised, and involved generating 16 possible symbols in 16 positions, or *houses*. Each of these symbols and houses have a particular meaning, and so the combination of house and symbol is intepreted by the practitioner in answering questions or making predictions. These symbols are generated by a random process involving the drawing of random lines, which are converted into (what we now call) binary symbols. From these random lines, everything else follows!

[1] The author of the paper also has a very interesting talk on the use of fractals in African art: https://www.ted.com/talks/ron_eglash_the_fractals_at_the_heart_of_african_designs

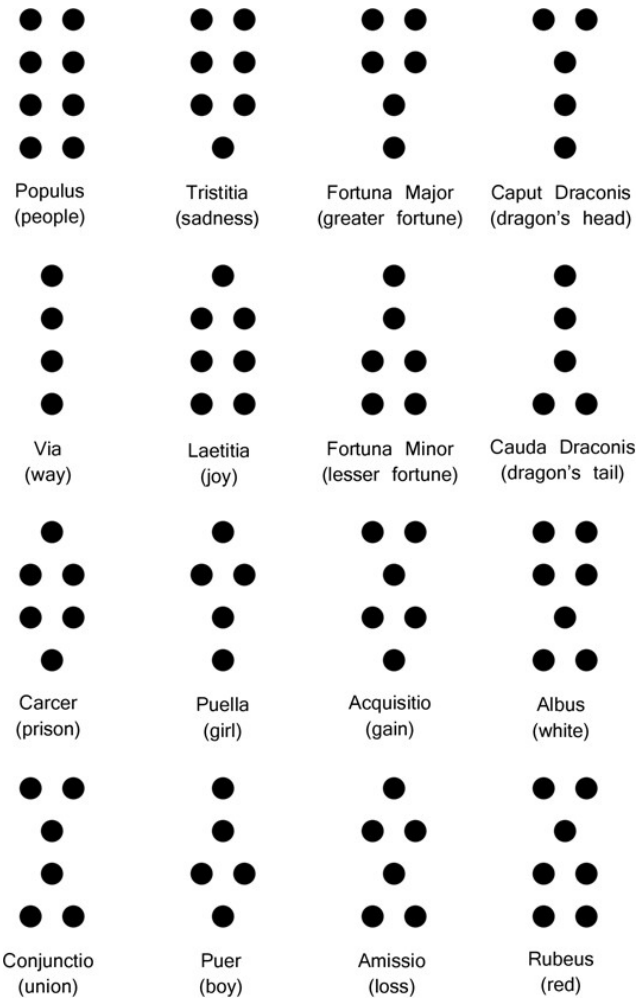

Figure 2: Illustration of the 16 possible geomancy symbols that can be generated. Notice how we can think of each symbol as a binary string: one dot is the number 1, while two dots is the number 0. The *rubeus* symbol, for example, can be encoded as the binary string 0100.

We will break down the procedure into a number of smaller subtasks (as is good programming practice), but at the end of the assignment we will be able to create a program that can take as input a question from a user, and generate all 16 symbols as well as their names. The interpretation of the results is, unfortunately, up to you!

Before beginning, I highly recommend watching the video on Moodle where we walk through the entire process of generating the symbols.

1 The First Task

The first part of the process involves the drawing of a random number of lines in the sand. These lines are then converted into a binary number/symbol and are then used later on in the ritual.

Therefore, to get us started, write a program that takes as input a single line of text. The line may consist of any number of spaces or dashes (-), but no other characters. Given this line, display whether there are an odd or even number of dashes.

1.1 Input

Input consists of a single line of text consisting of any number of spaces and dashes. You may assume that there will always be at least one dash character.

1.2 Output

If there are an even number of dashes in the line, output 0. Otherwise, output 1.

1.3 Example Input-Output Pairs

Sample Input #1

```
- - - - - - -
```

Sample Output #1

```
1
```

Sample Input #2

```
-- -- -   -- -- -
```

Sample Output #2

```
0
```

Sample Input #3

```
--
```

Sample Output #3

```
0
```

---

2 The Second Task

When generating symbols, the Bamana repeat the process mentioned in the previous task four times. Each line represents a binary digit, and four lines together make up a single symbol (consisting of four binary digits). See the figure below.

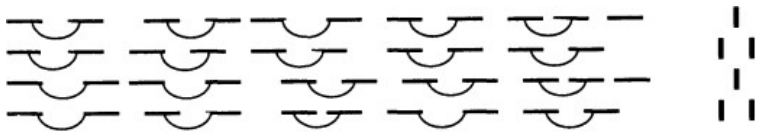

Figure 3: In the above example, four rows of dashes have been drawn. The symbol on the right is the binary encoding of these four lines (where 0 is represented by two vertical lines). The first row contains an odd number of dashes, the second row is even, the third odd and the fourth even. Therefore, the binary encoding for these sets of lines is 1010. This binary encoding is represented by vertical lines to the right of the dashes.

The above procedure generates a *single* geomancy symbol. All of this is then repeated a further four times to generate four symbols in total.

Write a program that reads in 4 sets of 4 dashed lines (for a total of 16 lines) and outputs the four binary symbols that each set of four lines represents.

2.1 Input

Input consists of 16 lines in total, consisting of any number of dashes and spaces. The first four lines represent the first symbol, the second four lines the next symbol, and so on. You may assume that there will always be at least one dash character on each line.

2.2 Output

Print out the four binary-encoded symbols represented by the 16 lines in total. Each binary symbol should be on its own line.

2.3 Example Input-Output Pairs

Sample Input #1

```
-  -  -
- - - -
- - - --
-- -- - -
-  -  -
- - - -
- - - --
-- -- - --
-  -  -
- - - -
- - - ---
-- -- - -
-  -  --
- - - -
- - - ---
-- -- - --
```

Sample Output #1

```
1010
1011
1000
0001
```

3 The Third Task

Having generated four symbols in the previous step, the next phase involves combining the symbols to generate two new symbols. See the figure below.

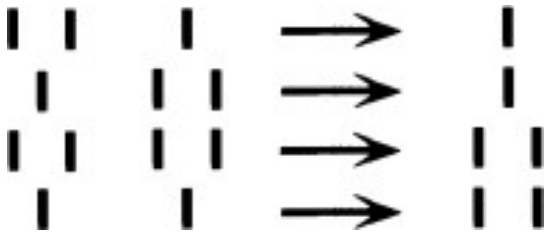

Figure 4: In the above example, we have two symbols which we can represent by the binary strings 0101 and 1001. These are combined by counting the number of lines along the rows: the first row has 3 lines and so the resulting symbol's first value is 1. The second row also has three lines, while the remaining two have an even number. The resulting symbol is therefore 1100.

Write a program that reads in two binary-encoded symbols, and outputs the resulting symbol when they are combined together.

3.1 Input

Input consists of two lines, representing two symbols. Both lines consist of four binary digits only.

3.2 Output

Display a single binary-encoded symbol produced by combining the two inputs.

3.3 Example Input-Output Pairs

Sample Input #1

1010
1011

Sample Output #1

0001

Sample Input #2

0010
1010

Sample Output #2

1000

Sample Input #3

0101
1010

Sample Output #3

1111

---

4 The Fourth Task

In the second task, we generated four symbols from a set of lines. Skipping ahead in the procedure, these four symbols are used to generate four more symbols. To do this, they are lined up and then read "sideways" — since each symbol is drawn as four rows, this generates a further four symbols (see the figure below).

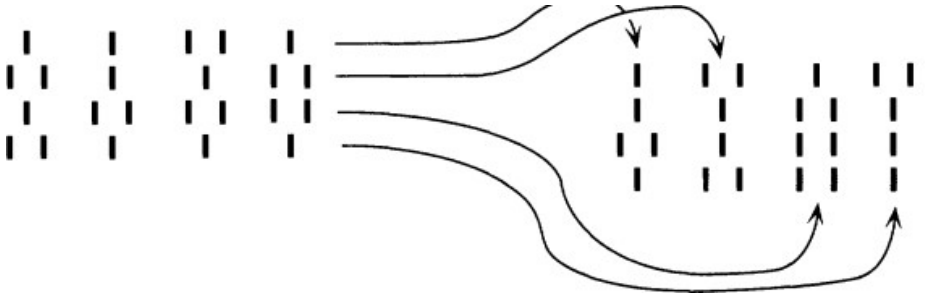

Figure 5: On the left are the four initial symbols, lined up next to one another. To generate a further four, we simply turn each entire row sideways! In this example, the first row consists of two single lines, a double line and a single line. The first new symbol therefore consists of the same, which can be encoded as 1101 in binary. Similarly, the second row contains a double line, two singles and then a double. The resulting symbol is therefore 0110. This is repeated for all four rows.

Write a program that reads in four binary-encoded symbols, and outputs four new symbols created by reading the resulting symbols "sideways".

4.1 Input

Input consists of four lines, representing four symbols. All lines consist of four binary digits only.

4.2 Output

Display all four binary-encoded symbols produced by reading the entered symbols sideways.

4.3 Example Input-Output Pairs

Sample Input #1

```
1010
1101
0101
1001
```

Sample Output #1

```
1101
0110
1000
0111
```

Sample Input #2

```
0101
1001
1101
1111
```

Sample Output #2

```
0111
1011
0001
1111
```

Sample Input #3

```
1111
0000
1010
0101
```

Sample Output #3

```
1010
1001
1010
1001
```

---

5 The Fifth Task

Given everything that has come before, we are now ready to generate all 16 symbols! In this task you are required to read in the first four symbols as binary encoded strings, and output all 16 binary-encoded symbols. To do this, we follow the process outlined below:

1. Read in the first four symbols. We will refer to these symbols as A, B, C, D.
2. The next symbol (which we will call E) is generated by combining A and B as in the Third Task.
3. The next symbol (which we will call F) is generated by combining C and D as in the Third Task.
4. The next symbol, G, is generated by combining E and F.
5. Next, we generate four more symbols by reading the symbols A,B,C,D "sideways", as we did in the Fourth Task. We will refer to these new symbols as H,I,J,K.
6. Once more, the next symbol, L, is generated by combining H and I.
7. A new symbol, M, is generated by combining J and K.
8. Another symbol N is generated by combining L and M.
9. The second-last symbol, O is generated by combining the symbols G and N.
10. The final symbol, P is generated by combining O with the very first symbol A.

5.1 Input

Input consists of four lines, representing four symbols. All lines consist of four binary digits only.

5.2 Output

Display all 16 symbols generated by following the above process. The symbols should be printed in the order they are generated according to the above rules (i.e. in the order A–P). Each symbol should be printed on its own line.

5.3 Example Input-Output Pairs

Sample Input #1

```
1010
1101
0101
1001
```

Sample Output #1

```
1010
1101
0101
1001
0111
1100
1011
1101
0110
1000
0111
1011
1111
0100
1111
0101
```

Explanation

The first four lines of output are the symbols A,B,C,D, which are exactly those given as input. The remaining lines are the symbols E–P printed in the order they are generated.

6 The Sixth Task

Each of the symbols generated by the process in the Fifth Task takes one of 16 shapes (since they consist of 4 binary digits, and $2^4 = 16$). Every possible shape is given a name, indicated in the table below.

Symbol	Name
1111	The Way
1110	Tail of the Dragon
1101	The Boy
1100	The Lesser Fortune
1011	The Girl
1010	Loss
1001	The Prison
1000	Joy
0111	Head of the Dragon
0110	The Conjunction
0101	Gain
0100	Red
0011	Greater Fortune
0010	White
0001	Sorrow
0000	People

In addition to this, each symbol belongs in a particular house. However, because of the Arabic origin of geomancy, the houses are numbered from right to left, and so the house number of a particular symbol does not correspond to the order in which that symbol was generated. In Figure 6 on the next page, we see the symbols (labelled A–P as per the previous task) together with their house number.

Furthermore, each house also has a name! The name of each house is given below.

House Number	Name
1	Life
2	Riches
3	Brothers
4	Father
5	Sons
6	Health
7	Spouse
8	Death
9	Journeys
10	Kings
11	Good Fortune
12	Prison
13	Witness 1
14	Witness 2
15	Judge
16	Reconciler

Write a program that accepts 16 generated symbols as binary encodings, and outputs a table listing each house, its name and the name of the symbol in that house.

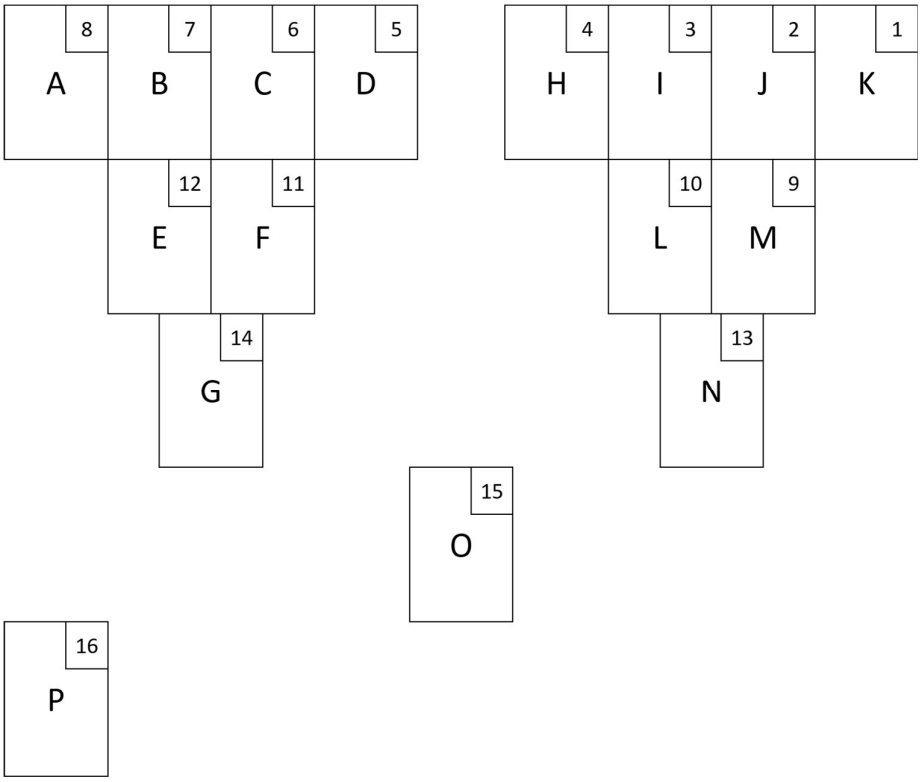

Figure 6: The layout of the symbols and their house numbers. Symbols are generated in the order A, B, C, ..., P. The top right corner of each block indicates the symbols associated house. For example, symbol A is in house 8, while symbol K is in house 1.

6.1 Input

Input consists of 16 lines, representing 16 symbols. All lines consist of four binary digits only.

6.2 Output

The first line of output should be the strings House, Name and Symbol. House and Name should be separated by two tab characters (be sure to use the tab character \t and not multiple spaces), and Name and Symbol by two tab characters. The next 16 lines should each display the house number, the name of the house, and the symbol associated with that house. On each of these lines, the house number, house name and symbol should be separated by several tab characters. There should be two tab characters between the house number and house name. However, the number of tab characters between the house name and the symbol name depends on the length of the house name and is given in the table below.

House name length	Tabs between house name and symbol name
8 or more	1
Less than 8	2

6.3 Example Input-Output Pairs

Sample Input #1

```
1010
1101
0101
1001
0111
1100
1011
1101
0110
1000
0111
1011
1111
0100
1111
0101
```

Sample Output #1

House	Name	Symbol
1	Life	Head of the Dragon
2	Riches	Joy
3	Brothers	The Conjunction
4	Father	The Boy
5	Sons	The Prison
6	Health	Gain
7	Spouse	The Boy
8	Death	Loss
9	Journeys	The Way
10	Kings	The Girl
11	Good Fortune	The Lesser Fortune
12	Prison	Head of the Dragon
13	Witness 1	Red
14	Witness 2	The Girl
15	Judge	The Way
16	Reconciler	Gain

7 The Seventh Task

The final task is to put everything together to make one big program that reads a question from the user and outputs all 16 symbols pictorially, as well as the associated table from the previous task.

To do this, download the code provided and complete the specified functions to create a program capable of doing so. **You may add as many additional functions as you wish, but do not modify any of the existing code.**

The functions to be implemented are as follows:

- markings_to_symbol: this function accepts four lines of markings (where each line is consists of any number of dashes and space characters) and returns a binary symbol as a string, similarly to the Second Task.
- generate_below: this function accepts a list of four binary symbols. The function should return a list of three binary symbols as strings. If the input consists of symbols A,B,C,D, then the function must combine A with B to form E, C with D to form F, and E with F to form G. The returned symbols should therefore be the list containing E,F,G.
- rotate: this function accepts a list of four symbols and generates four more symbols by reading them "sideways", as in the Fourth Task. The function should return a list of these four symbols.
- generate_judge: this function is responsible for generating the 15th symbol. It accepts two binary symbols and returns a single string representing the symbol formed when the two are combined.
- generate_reconciler: this function is responsible for generating the 16th symbol. It accepts a list of all the fifteen symbols generated so far, and returns a single string representing the symbol formed when the very first symbol and very last (15th) symbol are combined.
- show_table: this function accepts the list of 16 symbols in the order they were generated. The function does not return anything, but it prints out the table specifying the house numbers, names and symbols in the exact same format as the Sixth Task.
- reorder_symbols: this function accepts the list of 16 symbols in the order they were generated. It then reorders them to create a *list of lists*. Each element in the outer list represents a single row, where each row contains the symbols belonging to that row. The function retirns this list of lists. See the diagram below for more information on which symbols are expected in which rows.

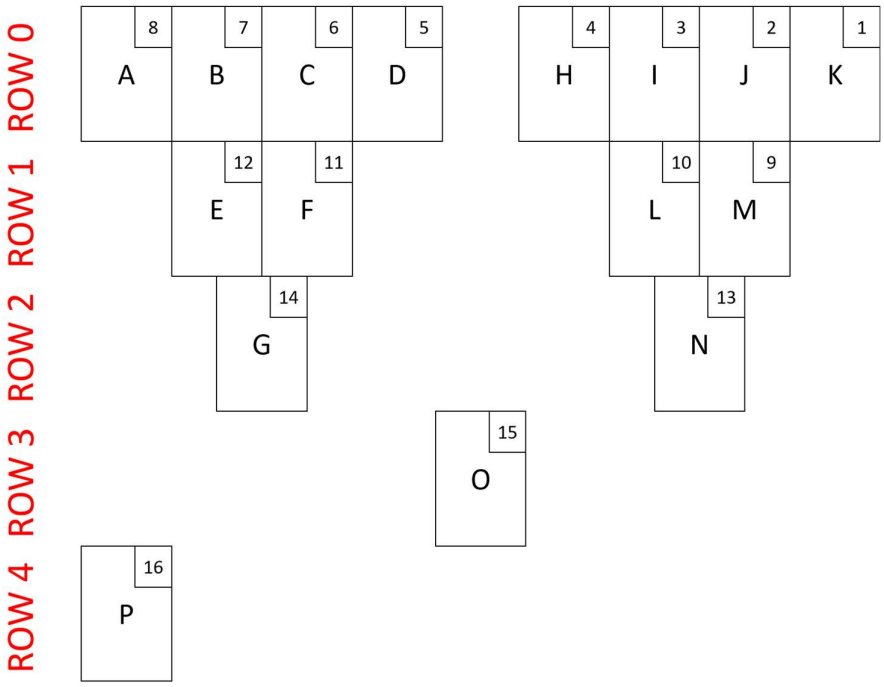

Figure 7: The red annotations indicate the rows that each symbol belongs to. For example, row 0 should contain the symbols A,B,C,D,H,I,J,K in that exact order, while row 4 contains only the symbol P.

7.1 Input

A single string representing the question the user wishes to ask.

7.2 Output

Output consists of the 16 symbols arranged in the correct order, followed by the table specifying the house and symbol information (see below for an example).

7.3 Example Input-Output Pairs

Sample Input #1

Will Kaiser Chiefs ever win the PSL?

Sample Output #1

House	Name	Symbol
1	Life	The Boy
2	Riches	The Way
3	Brothers	The Prison
4	Father	Joy
5	Sons	Head of the Dragon
6	Health	White
7	Spouse	Greater Fortune
8	Death	The Way
9	Journeys	White
10	Kings	Sorrow
11	Good Fortune	Gain
12	Prison	The Lesser Fortune
13	Witness 1	Greater Fortune
14	Witness 2	The Prison
15	Judge	Loss
16	Reconciler	Gain