PROBLEM 1 – Goldilocks is Scrambled!

In the story Goldilocks and the Three Bears, each bear had a bowl of porridge to eat while sitting at their favourite chair. What the story didn't tell us is that Goldilocks moved the bowls around on the table, so the bowls were not at the right seats anymore. The bowls can be sorted by weight with the lightest bowl being the Baby Bear's bowl, the medium bowl being the Mama Bear's bowl and the heaviest bowl being the Papa Bear's bowl.

Write a program that reads in 10 different sequences of three weights (a triad) and prints out the weight of Mama Bear's bowl from each triad. The first number of each input sequence is the number of triads to expect (so in the sample below, there are 2 triads to read in from the first sequence of numbers, and 5 triads from the second). You may assume all weights are positive integers less than 100 and that no triads will include the same number twice.

DATA11.txt (DATA12.txt for the second try) will contain 10 cases. You can assume that all input data will be valid. Your program must output the correct weight from each triad. So, you should have a total of 10 lines after running your program on the data.

Sample Input

```
2
10
5
8
9
2
4
5
28
12
31
9
10
19
39
20
11
45
33
21
0
19
18
```

Sample Output

```
8 4
28 10 20 45 18
```

PROBLEM 2 – Calendar

Write a program to print out a calendar for a particular month given the day on which the first of the month occurs together with the number of days in the month. This question is a lot about formatting, so your formatting must be PERFECT to receive full marks.

Your program should read pairs of integers. First is the integer representing the day of the week on which the month begins (1 for Sunday, 2 for Monday, ..., 7 for Saturday), and second is an integer which is the number of days in the month (only between 28 and 31 inclusive). Your program should print the appropriate calendar for the month.

DATA21.txt (DATA22.txt for the second try) will contain 20 lines. The data will be in pairs – the first number is the day of the week and the second is number of days. You can assume that all input data will be valid. Your program must output the correct monthly calendar for each situation, producing a separate calendar for each pair of numbers. So, you should have a total of 10 calendars after running your program on the data. Note that your output should be in the font "Courier New". If necessary, the output can be copied/pasted into a text editor and changed to Courier New for judging.

Sample Input

```
3
30
7
28
```

Correct Sample Output

```
Sun Mon Tue Wed Thr Fri Sat
          1   2   3   4   5
  6   7   8   9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30

Sun Mon Tue Wed Thr Fri Sat
                        1
  2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28
```

Incorrect Sample Output – these would **NOT** receive marks because of formatting issues:

Input 7 28

```
Sun Mon Tue Wed Thr Fri Sat
                        1
    2   3   4   5   6   7   8
  9  10  11  12  13  14  15
 16  17  18  19  20  21  22
 23  24  25  26  27  28
```

Input 3 30

```
Sun Mon Tue Wed Thr Fri Sat
          1  2  3  4  5
  6  7  8  9  10  11  12
 13  14  15  16  17  18  19
 20  21  22  23  24  25  26
 27  28  29  30
```

PROBLEM 3 – A Little Word Play

Mr. Swaine has an activity in his class where students make WordArt, where a piece of art is made using words that means something to them. Now he's combining programming with this idea, and he wants you to make a program that will format a piece of art!

The idea is quite straightforward … the user just picks a visible character of their choice, along with a word. The program takes the input and creates a square where the word is always printed diagonally in both directions (top left to bottom right, and bottom left to top right), and *if* the length of the word is odd, the word is also printed straight up/down in the middle of the square and straight left/right in the middle of the square. The rest of the square is filled with the chosen character.

DATA31.txt (DATA32.txt for the second try) will contain 20 lines. The data will be in pairs – first the character to fill the square with, second the word to work with. Your program must output the correct word art for each situation, producing a separate piece of art for each case. So, you should have a total of 10 pieces of art after running your program on the data. You can assume that all words will have a length above zero but below 20 with no spaces.

Sample Input

```
-
BASEBALL
.
HIGHLANDS
```

Sample Output

```
B------L
-A----L-
--S--A--
---EB---
---EB---
--S--A--
-A----L-
B------L

H...H...S
.I..I..D.
..G.G.N..
...HHA...
HIGHLANDS
...HAA...
..G.N.N..
.I..D..D.
H...S...S
```

PROBLEM 4 – Trucker Rules

A truck driver is planning to drive along the Trans-Canada highway from Vancouver to St. John's, a distance of 7000km, stopping each night at a motel. The driver has been provided with a list of locations of eligible motels, with the respective distance of each motel, measured in km, from the starting point in Vancouver. Some of the motel locations are:

0,990,1010,1970,2030,2940,3060,3930,4060,4970,5030,5990,6010,7000

… but more motel locations may be added just before the trip begins.

Determine if it is possible to complete the journey if:
1. the trucking company insists that the driver travels a minimum distance of A. km per day,
2. the law sets a maximum distance of B km per day, and
3. each night, the driver must stay at an eligible motel (from the above list or the additional locations described below).

The driver is interested in different options when making the trip, and you are to write the program to calculate how many different options there are.

For example, if no new motel locations are added, A = 1 and B = 500, then it is impossible to make the trip, i.e., the number of options is 0. If A = 970 and B = 1030 then there is one way to make the trip, but if A= 970 and B = 1040 then there are four ways to make the trip. There are two ways to make the trip if A= 970, B = 1030, and we add one stop at 4960.

Input Specification
There are 10 different situations in the file DATA41.txt (DATA42.txt for the second attempt). The first two lines are the minimum distance A and the maximum distance B (0 <= B <= 7000). Both A and B are integers. The third line of the input is an integer N between 0 and 20, followed by N lines, each giving the location m of an additional eligible motel (0 < m < 7000). You should note that no two motels are located at the same distance from Vancouver.

Output Specification
Output one integer - the number of different ways the driver can choose the motels to make the trip, under the given constraints.

Sample Input

```
1
500
0
970
1030
0
970
1040
0
970
1030
1
4960
```

Sample Output

```
0
1
4
2
```

PROBLEM 5 – Irregular Expressions

You might have heard of Regular Expressions, but this question is about Irregular Expressions, a concept we just invented. An irregular expression is a pattern string that can be used to match target strings.

Here's an example pattern that can be used to "accept" or "reject" target strings:

adbsk[kkjsvf]bbb

There are two rules for matching a target string to a pattern like the one above:

1. Any lower-case letter that is not inside square brackets in the pattern must match to the same letter in the target string, in the correct position.
2. If a set of lower-case letters appears in square brackets in the pattern, the target string must contain, at that point in the string, exactly half of these letters (rounded up) in any order.

There are lots of target strings that match the above pattern (60 to be precise). Here are a few of them. The parts that match the bracketed portion are highlighted:

adbskkkjbbb, adbskjkkbbb, adbskskfbbb, adbskfjkbbb

Of course, there are also an infinite number of targets that do not match. Here are a few of them:

adbskkkjsbbb, adbskjkbbb, adbsjsvbbb, adbsksfjbb, fhfghj tomato pesto

An irregular expression can be any length, can use any plain lower-case letter, and can contain zero or more non-nested square bracket sections.

DATA51.txt (DATA52.txt for the second try) will contain 10 lines. Each line will consist of an irregular expression pattern string followed by 5 target strings to match, all separated by spaces. Your program must output either "true" or "false" for each target string depending on whether it matches the pattern. The words "true" and "false" must be lower case and separated by a single space. You should produce a separate line of text for each line in the input. The maximum length for each pattern and each target string is 256 characters. Note that the test data below has only 5 lines, but the real data file will have 10.

Sample Input

adbsk[kkjsvf]bbb adbskkkjbbb adbskkkjsbbb adbskskfbbb adbsjsvbbb fhfghj
fkqm[qzqyledbqq]c fkqmdqqqlc fkqmqlbyq fkqmqedqc fkqmblqqzc fkqmlqbdzc
dj[mocn]dd djnmdd djocdd djmodd djmodd djcndd
wsvahfskbs[uucysmlfrcnhu]mjgphbd this problem is ridiculous dude
a[aauzujbtipmca]bsk aiamzuausk acuaaujabsk aaaauucbsk aazpubtubsk aiaujcambsk

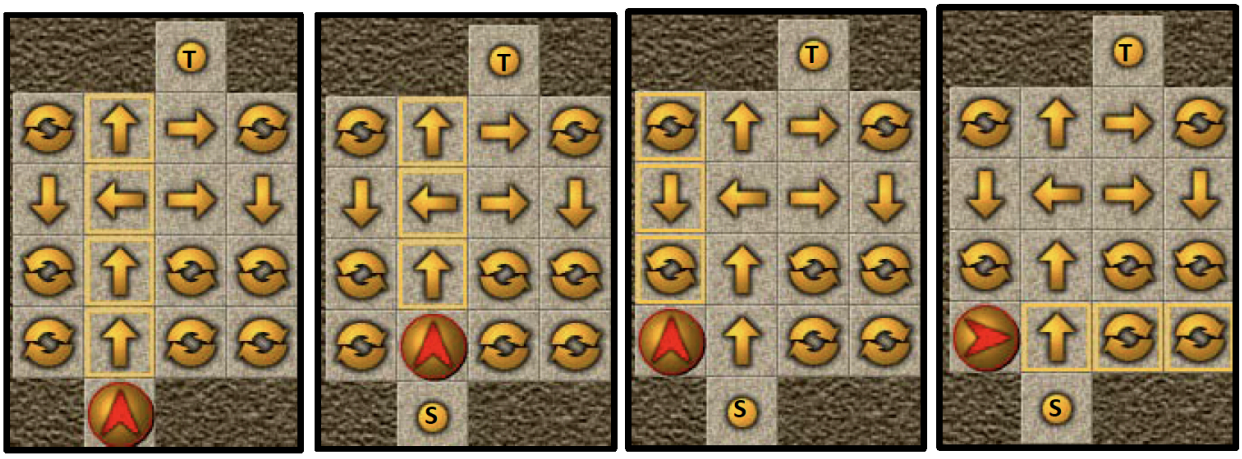
Sample Output

true false true false false true false
false true true true true true true
true false false false false false
false true false true true

PROBLEM 6 – What Lies Ahead

You are trapped in a room with a puzzle grid painted on the floor. You are in a start square, just below the puzzle grid, facing up. Your task is to get to a target square (or suffer a terrible fate). Your first move must be to step into the main puzzle grid if you can. Once in, you may not step outside the grid unless it is onto a target square. Once you step onto a target square, you are finished and can make no further moves.

Each square in the puzzle grid has a symbol painted on it representing a possible move (up, down, left, right, clockwise, or counterclockwise). You can only move one grid square at a time, and you can only turn 90 degrees at a time. Your next move must always be one of the moves you can see painted on the floor ahead of you (this does not include the one you are standing on). You cannot change the direction you are facing unless you can see a move ahead that allows you to turn clockwise or counterclockwise. Because of the restrictions on the direction you can face, you will often end up stepping sideways or backwards when completing this puzzle.



In the sequence of moves shown above, the user starts at the bottom, facing up. Her target is the square above the maze at the top. For her first move, the choices are UP or LEFT (just the moves she sees in front of her). She can't go left, so she chooses UP. Then her choices are UP or LEFT again and she chooses LEFT. Now her choices have changed: CLOCKWISE, COUNTERCLOCKWISE, or DOWN. She can't move down and if she turns counterclockwise, she will have no moves in front of her. So, she chooses CLOCKWISE. For her next move, she can now choose from UP or CLOCKWISE.

DATA61.txt (DATA62.txt for the second try) will contain 10 test cases. Each test case consists of 6 lines of 6 characters, representing a 4x4 puzzle grid as well as all possible start and target squares around the outside of the grid. The moves are U, D, L, R, C, and B for UP, DOWN, LEFT, RIGHT, CLOCKWISE, and COUNTERCLOCKWISE respectively. The start square is marked S and can be anywhere on the bottom row. There will be five possible target squares, marked T, which can appear anywhere around the outside of the puzzle grid. Squares which are out of bounds are marked with the "." character (ASCII code 46).

Write a program to determine which of the target squares can be reached from the given start square. You should output a single line of 5 characters, with each character representing a target square (in order from top left, scanning each row from left to right, moving downwards). If a target square is reachable, output T for that square. If not, output F.

Note that in the sample input below, there are 2 test cases, but the real data files will contain 10. The squares which are part of the puzzle grid are shaded below for ease of reading, but this shading will not appear in the file.

Sample Input

```
..TT..
TCURC.
.DLRD.
.BUBB.
.CUCCT
.TS...
.T.T..
TRCDU.
.UCLDT
.RCBL.
TDUCU.
.S....
```

Sample Output

```
TTFFT
FTFFF
```