ACM Programming Contest - All Problems

Dixie State University

March 28, 2015

Do NOT turn this page until

10:00 am Mountain Time!

Rules

1. Each team consists of up to three students.
2. Each team may use one provided computer.
3. Teams may use any printed references.
4. Teams may not use any electronic aids, including the internet. The help system built in to your programming environment is okay. Official documentation websites such as docs.python.org are okay.
5. University, high school and beginner team categories compete only within their category.
6. All solutions must be written in C++ or Java. Python is also permitted for high school and beginner teams, and for university teams where no member has completed CS 3005 or any other course that requires C++.
7. The team with the most correct solutions wins.
8. Ties will be broken using a time score:
   (a) Each time a team submits a correct solution, the number of minutes that have elapsed since the beginning of the competition is added to the time score.
   (b) For each incorrect solution submitted, a 20 minute penalty is added to the time score, but only if the team eventually submits a correct solution to that problem.
   (c) Multiple penalties will be added for multiple incorrect solutions to the same problem.
9. The input for each problem comes from standard input. This goes by the names `cin` and `STDIN` in C++, `System.in` in Java, and `sys.stdin` in Python. Other methods may also exist for receiving input from standard input for each language.
10. The output for each problem should be sent to standard output. This goes by the names `cout` and `STDOUT` in C++, `System.out` in Java, and `sys.stdout` in Python. Other methods may also exist for sending output to standard out for each language.
11. The output of submitted solutions must exactly match the output of the reference solution, down to the last character. Whitespace differences matter. Any other output, including debugging output, may cause an otherwise correct solution to be marked as incorrect. Each problem statement with example input and output shows exactly where newline characters are placed and where spaces are appropriate.
12. Solutions have a 10 second time limit. Any solution that runs longer than that will be considered incorrect.
13. The university and high school contests begin at 10:00 am and ends at 3:00 pm. The beginner contest begins at 12:00 pm and ends at 3:00 pm.

Contents

1 Sample Problem

Write a program to read a number N and a word, then repeat the word N times.

The first line of input the program will be a number $1 \leq N \leq 100$, which specifies how many times to repeat the word. The second line of input will contain a single word.

The output from the program will contain N lines. Each line will contain the word read from the input.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
5↵
wumpus↵
```

Sample Output

```
wumpus↵
wumpus↵
wumpus↵
wumpus↵
wumpus↵
```

2 Box

This program must draw a solid filled box using the * character.

The input to the program will be two numbers on the same line. The first number $3 \leq W \leq 9$ is the width of the box. The second number $3 \leq H \leq 9$ is the height of the box.

The output of the program will have H lines, each with W * characters.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
4 6↵
```

Sample Output

```
****↵
****↵
****↵
****↵
****↵
****↵
```

3 Death Valley

A town in Death Valley has a water tank that holds 10,000 gallons of water. Assume that the water tank is full. Calculate the number of weeks the town will be able to use water if no more water is put into the tank.

The input to the program will be a number on each line $1 \leq G \leq 10000$, the number of gallons that the town uses per week. The last line will have $G = 0$ and should not be processed.

The output of the program will have one line per input line. Each line will display the number of full weeks the town can use water at the given rate.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
1750↵
1000↵
4325↵
0↵
```

Sample Output

```
5↵
10↵
2↵
```

4 Spelling Bee

Write a program to help the judges at a spelling bee. You will check to see if two words are spelled the same, and report the result.

The first line of input the program will be a number $1 \leq N \leq 100$, which specifies the number of checks to make. There will follow N lines, with two words on each line, separated by a space.

The output from the program will contain N lines. The each line will either say "WORD1 and WORD2 are the same", or "WORD1 and WORD2 are different", depending on whether the words match or not. WORD1 and WORD2 should be replaced with the words to be compared.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
3↵
FRED BARNEY↵
WINDOWS WINDOWS↵
LINUX LINUS↵
```

Sample Output

```
FRED and BARNEY are different↵
WINDOWS and WINDOWS are the same↵
LINUX and LINUS are different↵
```

5 Rock Paper Scissors

Write a program to determine the winner of a series of games of Rock-Paper-Scissors. In this game, Rock wins Scissors, Scissors wins Paper and Paper wins Rock. If both players choose the same option, the game is a tie.

The first line of input the program will be a number $1 \leq N \leq 100$, which specifies the number games to judge. There will follow N lines, with two words on each line, separated by a space. Each word will be Rock, Paper or Scissors. Player A is the first word, and player B is the second word.

The output from the program will contain N lines. The each line will say "A wins", "B wins", or "tie".

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
3↵
Rock Paper↵
Paper Rock↵
Scissors Scissors↵
```

Sample Output

```
B wins↵
A wins↵
tie↵
```

6 Factorial

If N is an integer number greater than 0, the factorial of N is defined as $N \times (N-1) \times (N-2) \times (N-2) \times ... \times 1$.

For example, the factorial of 4 is $4 \times 3 \times 2 \times 1 = 24$.

Write a program to calculate the factorial of positive integers.

The first line of input to the program will be a number $1 \leq K \leq 20$. This will be followed by K lines. Each of the lines will contain one integer, $1 \leq N \leq 12$.

The output of the program will have K lines, one for each N in the input. The output will contain the factorial of N.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
3↵
4↵
1↵
12↵
```

Sample Output

```
24↵
1↵
479001600↵
```

7 Melody

Max and Greta are exo-linguists traveling the galaxy learning languages. They are building a translator application to help Earthlings communicate with the people of Giwdul.

In the Giwdul language, each word is a melody of up to 16 notes. The actual notes in a word aren't as important as whether each note has a frequency lower, the same, or higher than the previous note in the word. For example, if two consecutive notes have frequencies of 440 and 698 this is an Up transition. Another pair of notes may have frequencies of 587 and 622, which is also an Up transition. While another pair of notes may have frequencies of 622 and 440, which is a Down transition.

To translate from a sequence of frequencies into a sequence of characters to use in a dictionary, we use U for up transitions, D for down transitions, R for repeat, and * for the first note in the word. For example, given a word with these 7 notes: 493 466 830 830 830 659 659 we would spell the word *DURRDR.

Max and Greta already have a machine that listens to a melody and records the frequencies in the word as numbers. They also have a dictionary that translates the character words, like *DURRDR into English. Help them complete the task of creating a translator by writing a program that receives a list of frequencies and translates it into a list of characters as described above.

The first line of input to the program contains a single integer $1 \leq N \leq 1000$ that specifies the number of lines remaining in the input. All other lines specify a single word. These lines begin with a number $1 \leq M \leq 16$ that specifies the number of notes in the word, followed by M numbers. The numbers on the line are separated by spaces.

The output for the program contains N lines, with one set of characters per line. The characters are those created using the translation process described above.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
4↵
7 493 466 830 830 830 659 659↵
16 698 784 493 493 493 659 740 659 659 440 440 440 587 740 740 830↵
2 554 523↵
10 784 622 622 622 466 659 466 740 784 784↵
```

Sample Output

```
*DURRDR↵
*UDRRUUDRDRRUURU↵
*D↵
*DRRDUDUUR↵
```

8 Topswops

Maddison and Gata are reading extra material on permutations for their Discrete Mathematics course. They came across an interesting card game from the 1970s, called topswops. They want to understand it better, so they want to be able to run simulations of the game for many different situations.

Topswops is played with a deck of cards numbered sequentially from 1 to N. The cards are randomly shuffled. Then, the top card's number is examined, let's call it M. The top M cards are removed from the deck and placed in reverse order, then added to the top of the deck. The game repeats until the card numbered 1 reaches the top of the deck.

For example, in a 5 card game, the shuffle may produce a deck with cards ordered 3 4 2 5 1. A single turn would extract the top 3 cards and reverse their order, putting them back on top to produce the deck 2 4 3 5 1.

Help Maddison and Gata with this problem by creating a program to count the number of turns for a topswops game to complete given the initial order of the deck.

The first line of input will contain the number of cards in the deck $1 \leq N \leq 20$. The second line of input will contain the number of decks to solve $1 \leq P \leq 1000$. The following P lines will contain N numbers separated by spaces. Each of these is the shuffled order of the deck.

The output will contain P lines with one number per line. This is the number of turns until the 1 card moves to the top of the deck.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
5↵
4↵
3 4 2 5 1↵
2 5 4 1 3↵
1 4 3 2 5↵
5 2 4 3 1↵
```

Sample Output

```
4↵
6↵
0↵
1↵
```

9 Collatz Conjecture

Machiko and Gair are the first humans to return to the moon in 40 years. The astronauts discover rows of piles of rocks on the far side. Curious, they count the number of rocks in each of the piles in several rows. They discover that any pile with an even number of rocks is followed by a pile with half as many rocks. However, piles with an odd number of rocks are followed by a pile with one more than three times as many (that's $3 \times n + 1$). Each row ends with a pile of 1 rock.

For example, one row starts with a pile of 3 rocks. The next pile has 10 rocks, followed by 5, 16, 8, 4, 2, and finally 1 rock. So, there are 7 piles after the initial pile.

They get tired of counting the piles rocks and ask you to write a computer program that will take the number of rocks in the first pile of a row and then tell how many more piles there are in that row.

The input to your program is one positive integer per line, $1 \leq N \leq 1000000$ representing the number of rocks in the first pile of a row. The input will have no more than 100 lines. The last line will have $N = 0$ and should not be processed.

The output from your program is two numbers per line, separated by a single space. The first number is the number from the input, N. The second number is the count of how many more piles of rocks there would be on the row, not counting the first pile.

Note that some individual piles of rocks may have more than 2^{32} rocks in them.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
17↵
3↵
10↵
12↵
1↵
951652↵
0↵
```

Sample Output

```
17 12↵
3 7↵
10 6↵
12 9↵
1 0↵
951652 170↵
```

10 Biorhythms

Mason and Gene are learning all the culture of the 1970s from episodes of the TV show "CHiPs." They found a means of fortune telling called biorhythms. They decide to create an app that will allow people of today to receive this pseudo wisdom.

In the biorhythm theory, everyone has two cycles that begin on the day they are born. Each cycle has an associated power, ranging from 100 to -100. One's best days are those when both cycles have high positive powers.

The power, P, of a rhythm of length, L days, on a particular day of one's life, D, is calculated using this formula: $P = \lfloor 100 \times \sin(2\pi D/L) \rfloor$. In Python π can be found in `math.pi`. In C++ it can be found with `atan(1)*4`.

For example, for a rhythm with $L = 25$, on the 1010th day of your life ($D = 1010$), the power is $P = 58$. Note that the fractional part of the number is truncated, not rounded. Casting a floating point number to the `int` type will truncate it.

Mason and Gene already have a calculator for finding the day in someone's life, given their birthday, and a random number generator for picking the length of biorhythm cycles. Help them complete the app by calculating the rhythm powers for users, given $L1$, $L2$ and D.

The input to your program will begin with one line with two integer numbers, the length of the two cycles, $L1$ and $L2$. The rest of the input will be one integer number per line, the day in the user's life, D. The last line in the file will have $D = 0$. This line should not be processed. The inputs will obey these limits: $10 \leq L1, L2 \leq 100$, $0 \leq D \leq 50000$. There will not be more than 1000 values of D.

The output from your program will have one line per D value in the input. Each line will show three integer numbers, D, $P1$, and $P2$, with a single space between them and no space at the end of the line.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
28 23↵
1010↵
187↵
9842↵
0↵
```

Sample Output

```
1010 43 -51↵
187 -90 73↵
9842 0 -51↵
```

11 Pseudo Random Numbers

Martha and Goldie are excited to have been accepted as players on a game show where they get to choose briefcases with money in them and accept or deny the offer made to them by the host.

Practicing for the show, they are creating a computer program to help them play the game many times to understand the probabilities involved. They have most of the program planned, but they need a pseudo-random number generator.

A pseudo-random number generator is used to create a sequence of numbers that appear to have the properties of actual random numbers. The usual process is to start with an initial number, called the seed, and apply a mathematical transformation to generate the next number. Then the same transformation is used on the second number, generating another.

Early in the days of electronic computers, the middle-square method was used to generate pseudo-random numbers. In this method the seed number is squared and the middle digits of the square are extracted as the next number in the sequence.

For example, for 4 digit numbers, and a seed of 5234, the square is 27394756, an 8 digit number. The middle 4 digits are 3947, the next number. The following number in the sequence is 5788. If the squared number doesn't have enough digits, then add 0s on the left until it does. For example, if the seed is 0763, then the squared number is 582169. We add two 0s on the left to get 00582169, and the next number is 5821.

Help Martha and Goldie generate pseudo-random numbers for their program, using the middle-square method.

The input will contain 3 lines, with one number per line. The first number, N, is the number of digits that you want to have in your newly created pseudo-random numbers. N will be an even number in the range $2 \leq N \leq 8$. The second number, $1 \leq M \leq 1000$, is the number of pseudo-random numbers you need to generate. The third number, S, is the seed number.

The output will contain M lines, with one number per line, in the order created using the middle-square method. The first output number is the first number after the input seed. If the output number has leading 0 digits, do not display them. If the output number is 0, do display it.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
4↵
5↵
5234↵
```

Sample Output

```
3947↵
5788↵
5009↵
900↵
8100↵
```

Sample Input

```
2↵
4↵
62↵
```

Sample Output

```
84↵
5↵
2↵
0↵
```

12 Card Shuffle

Megan and Grant are considering careers as magicians. They have been able to acquire several broken card shuffling machines. Each shuffler produces the same shuffle every time. They plan to use the shufflers in a trick where they feed the same deck of cards through the shuffler until the top card in the deck is returned to the top. They just need to know how many times to feed the cards through for each different shuffler.

Help them plan for their trick by counting how many times a deck needs to be passed through a particular shuffler, until the top card returns to the top.

The first line of input to your program will be a single integer, N, the number of cards in the deck. The second line of input will be N numbers, indicating the new position of each card after 1 shuffle. The card positions are numbered 0 through $N-1$. The number of cards will obey this constraint: $1 \leq N \leq 100$.

For example, a shuffle description for 4 cards might be 3 1 0 2. This indicates the top card, with index 0 will be moved to the last position 3, the second card will stay in position 1, the third card will move to the top, or position 0, and the last card will move to position 2.

The output from your program will be a single line containing one integer number. This is the number of shuffles until the original top card returns to the top. Note that a single card deck still needs to be fed through the shuffler at least once for the trick.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
5↵
1 2 3 4 0↵
```

Sample Output

```
5↵
```

Sample Input 2

```
5↵
2 4 3 0 1↵
```

Sample Output 2

```
3↵
```

13 Devlali

Milo and Geoffrey are traveling through small towns in India. In Devlali, they find information about the mathematical concept of "generated numbers" and "self numbers".

To understand this idea take a number, such as 54. Add its digits to its value, $5+4+54 = 63$. Since 63 can be generated by this process, it is a generated number. However, 31 is a self number, because there is no number that can generate it with this process. Other numbers can be generated by more than one number. For example, $1 + 0 + 0 + 100 = 101$ and $9 + 1 + 91 = 101$. So, 101 is called a junction number.

Milo and Geoffrey want to apply this concept to large set of numbers, classifying each of them as self, junction, or generated numbers. Help them, by writing a program to determine the type of a number.

The first line of the input will have a single integer, $1 \leq N \leq 1000$, the number of lines that follow. For each following line, there will be one number, $1 \leq M \leq 10000$.

The output will consist of one line per input number M. The line will be M followed be a single space, followed by "self", "junction", or "generated". The word depends on the classification of M.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
6↵
63↵
31↵
101↵
9934↵
9993↵
9994↵
```

Sample Output

```
63 generated↵
31 self↵
101 junction↵
9934 junction↵
9993 self↵
9994 generated↵
```

14 Anagram Dictionaries

Milly and Gloria are avid word game players. Some of their favorites are Ruzzle and Words with Friends. They realize that these games can be played better if one understands anagrams.

An anagram is a word that is produced by rearranging the letters in another word. For example, ACT and CAT are anagrams, as are TORSO and ROOST.

Help Milly and Gloria create an anagram dictionary. This is produced by reordering each word's letters alphabetically, to produce an "alphabetical anagram". The alphabetical anagram for TORSO and ROOST is OORST. Each word that has the same alphabetical anagram is listed alphabetically on the line after the alphabetical anagram. The line for OORST would contain "OORST ROOST TORSO". The alphabetical anagrams are listed in alphabetical order in the anagram dictionary.

For example, if the words in our dictionary were ACT, BARE, BEAR, CAT, and FREE, the anagram dictionary would have lines for ABER, ACT, and EEFR. Where ABER would list BARE and BEAR as anagrams; ACT would list ACT and CAT as anagrams; and EEFR would list FREE as the anagram.

The input to your program will be a number, N, on the first line, followed by N lines, with one word per line. The number of words will be limited by $1 \leq N \leq 11000$. Each word will contain only upper case letters, A through Z.

The output of your program must be one line per alphabetical anagram, with the alphabetical anagram first, followed by a single space, followed by each of the anagrams, sorted alphabetically, each separated by a single space. There must not be a space at the end of the line. If the alphabetical anagram is itself a word, the word is repeated. The output lines must be sorted alphabetically by alphabetical anagram.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
5↵
ACT↵
BARE↵
BEAR↵
CAT↵
FREE↵
```

Sample Output

```
ABER BARE BEAR↵
ACT ACT CAT↵
EEFR FREE↵
```

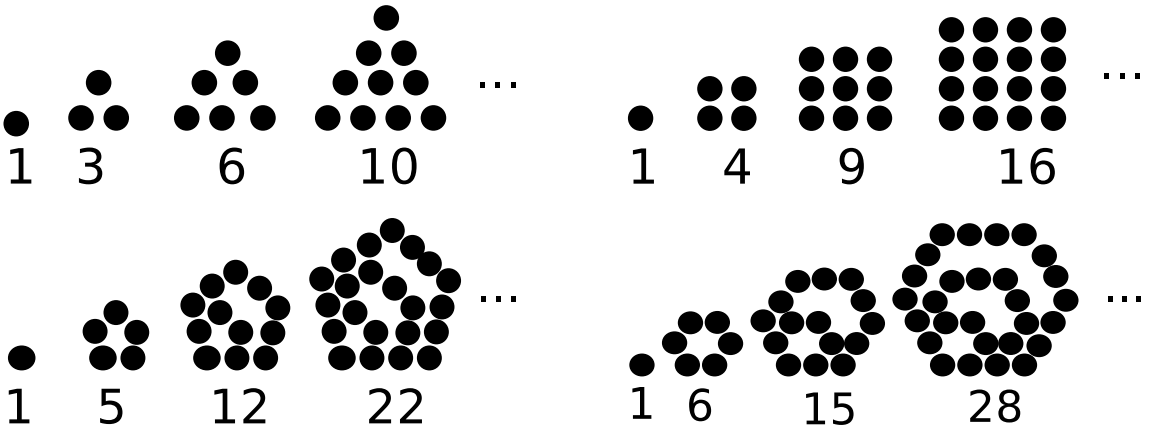
15 Figurate

Marge and Gunther are extraterrestrial archaeologists studying secret images sent back by the Mars rovers Spirit and Opportunity. They show markings like the following:

⬡⊙⊙⊙⊙⊙⊙▫⊙⊙⊙⬠⊙△⊙

It's a series of triangles, squares, pentagons, and hexagons, with each polygon followed by a number of ⊙s. To encode these markings in text files, Marge and Gunter write each collection of markings as numbers on a line. The preceding markings are recorded as 4 6 6 4 3 5 1 3 1. The first number describes the number of polygons in the marking. Each pair of numbers after that represent the number of sides in the polygon and the number of ⊙s that follow it.

Due to other evidence, they know that ancient Martians were fans of figurate numbers. These are numbers that can be found by counting the number of dots in regular polygons of various sizes. For example, the triangular, square, pentagonal and hexagonal numbers are found by counting the number of dots in each successively larger polygon.



Figurate numbers can also be calculated by summing a series. These are the formulas for the nth Triangluar, Square, Pentagonal, and Hexagonal numbers:

$T(n) = \sum_{i=1}^{n} i = 1 + 2 + 3 + ... + n$

$S(n) = \sum_{i=1}^{n} 2 * i - 1 = 1 + 3 + 5 + ... + 2 * n - 1$

$P(n) = \sum_{i=1}^{n} 3 * i - 2 = 1 + 4 + 7 + ... + 3 * n - 2$

$H(n) = \sum_{i=1}^{n} 4 * i - 3 = 1 + 5 + 9 + ... + 4 * n - 3$

Marge and Gunther interpret the Martian markings by treating each polygon followed by n ⊙s as the nth figurate number for the appropriate polygon. For example, ▫⊙⊙⊙ is interpreted as $S(3) = 1 + 3 + 5 = 9$, the 3rd square number.

Next, they add all numbers from a collection of markings. For example,
⬡⊙⊙⊙⊙⊙⊙▫⊙⊙⊙⬠⊙△⊙ is $H(6) + S(3) + P(1) + T(1) = 66 + 9 + 1 + 1 = 77$.

Finally, Marge and Gunter take consecutive runs of the numbers acquired by this process and treat them as ASCII values. They are amazed to find the resulting text reads as English

sentences.

Marge and Gunter want to interpret the large messages revealed in the images from Mars, but are tired of doing the calculations by hand. Help them by writing a program to decipher the ancient Martian messages. They had a bunch of students transcribe the markings into the format described here.

The first line of input consists of a number, $1 \leq K \leq 150,000$, which specifies the number of lines to follow. Each following line has the format of a number, $1 \leq M \leq 10$, followed by M pairs of numbers. Each of these pairs has a first number, P, and a second number, N. P describes the number of sides on the polygon, and N is the number in that sequence. These lines match the description given earlier in this document.

The program output must compute a single character from each of the K lines, using the process described above. These characters are concatenated into one string, then displayed verbatim, with a newline character following them.

In the first example below, note that last line indicates the third triangle number, 10. This is the ASCII code for the newline character. Thus, the output has the newline character described in the input, plus the following newline character.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
15↵
4 6 6 4 3 5 1 3 1↵
2 4 9 4 4↵
4 6 7 4 4 6 2 6 1↵
6 6 7 3 6 6 1 5 1 5 1 6 1↵
4 6 7 3 4 3 2 3 1↵
2 4 10 3 4↵
3 4 5 3 3 4 1↵
2 3 11 5 2↵
2 3 13 6 2↵
4 5 8 4 4 5 2 3 1↵
5 6 7 6 2 4 1 3 1 5 1↵
3 6 7 3 5 4 2↵
5 5 8 3 3 4 1 3 1 6 1↵
5 3 14 6 2 6 1 3 1 3 1↵
1 3 4↵
```

Sample Output

```
Martin Gardner↵
↵
```

Bonus Sample Input

```
13↵
4 6 6 4 2 5 1 3 1↵
2 4 10 4 1↵
3 6 7 4 4 6 1↵
4 6 7 6 3 3 1 6 1↵
4 5 8 5 3 6 2 6 1↵
3 3 7 3 2 3 1↵
4 4 8 3 2 4 1 3 1↵
3 4 9 3 5 5 1↵
5 3 14 6 2 5 1 4 1 5 1↵
4 3 14 6 2 6 1 4 2↵
3 3 13 5 3 6 1↵
3 3 7 4 2 5 1↵
1 3 4↵
```

Bonus Sample Output

The output for this sample is not given, but should make sense when you find it.

16 Pascal Triangle Art

Matilda and Gwyneth are creating an art collection for their first mathematical art show. They recently learned about a method to create pictures using a numerical sequence called Pascal's triangle. This sequence is usually written in the shape of a triangle. Here are the first five rows of Pascal's triangle:

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

Each row is created by adding neighboring pairs on the row above. For this purpose, you can imagine a temporary 0 on each side of the row.

It may be worth noting that the largest numbers in the Nth row of the triangle require N bits in a binary representation.

Help Matilda and Gwyneth by making an ASCII art image by making a program to create some number of rows in the triangle, replacing each number with text characters. The characters will be chosen based on whether the number is evenly divisible by a selected number.

The input will contain two lines, with one integer per line. The first number, $1 \leq N \leq 100$ is how many lines of the triangle to generate. The second number, $2 \leq M \leq 10$, is the modulus.

The output will contain N lines. Each line will be a representation of the row in Pascal's triangle, with the first row on top. The first row will be proceeded by $N - 1$ spaces; the second row by $N - 2$ spaces; and so forth. Each number in a row will be represented by two asterisk characters or two space characters, with no spaces between numbers. If the number in the row is not evenly divisible by M, then use the asterisks, otherwise use the spaces.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
8↵
2↵
```

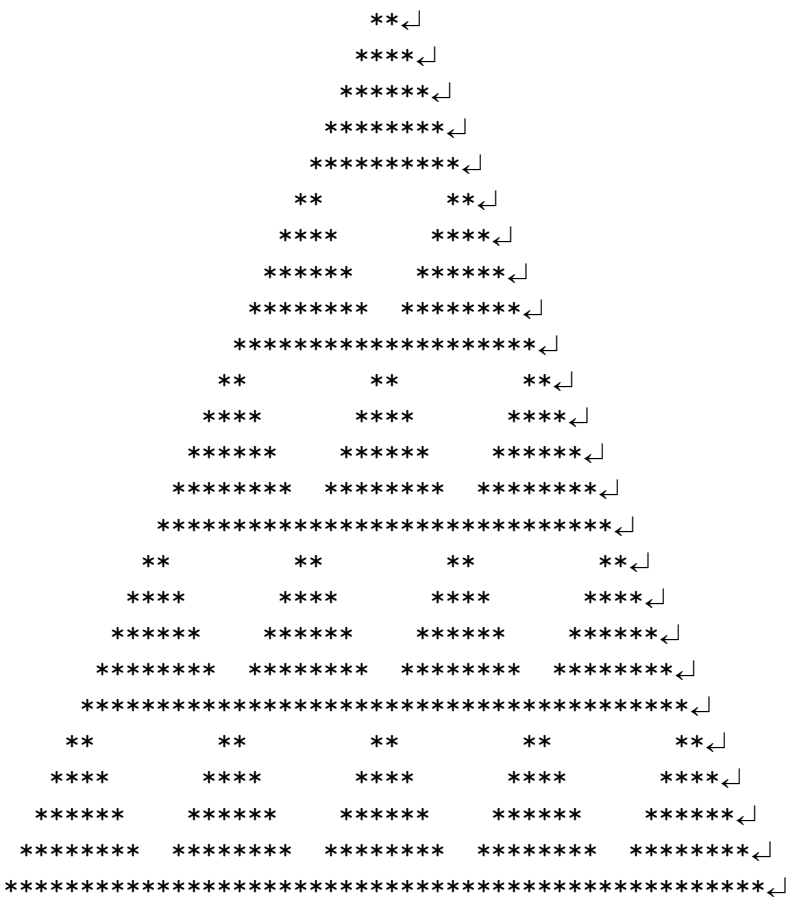
Sample Output

```
      **↵
    ****↵
  **  **↵
********↵
**      **↵
****    ****↵
**  **  **  **↵
**************↵
```

Sample Input

```
25↵
5↵
```

Sample Output



17 Penney

Maria and Gwen are vacationing in Paris, France. They are approached by a vendor on the street to play and bet on a two player coin flipping game, called Penney, after its inventor Walter Penney.

Each player chooses a pattern of three consecutive coin flips, from the 8 possible patterns: HHH, HHT, HTH, HTT, TTT, TTH, THT, and THH. (H = heads, T = tails). A coin is repeatedly flipped to generate a sequence of heads and tails. The first of the two patterns to show up wins.

For example, player 1 may choose HTH and player 2 may choose HHT. The coin is flipped until one of those patterns appear. After 6 flips the sequence HTTHTH is obtained, and player 1 wins. Notice the HTH at the end, and no HHT anywhere.

After listening to the game description, they see that the game is a confidence trick. So, of course, they agree to play, but demand that they choose their pattern after the vendor chooses.

Help Maria and Gwen win this game by writing a program that allows them to choose which of two triplets has a better percentage of winning. You may want to use a game simulation to calculate the results of the $8 \times 8 = 64$ possible comparisons.

The first line of input will be a single number $1 \leq N \leq 64$, the number of problems to follow. There will be N more lines, each with one problem. Each problem will have two 3-letter patterns of H and T, separated by a space.

The output will have N lines, one for each problem in the input. For each problem the output will be the pattern that has the higher probability of winning this game. If both patterns have equal probability, the output should be the word TIE.

Note: The ↵ symbol in the examples below represents a newline character.

Sample Input

```
4↵
HHT THT↵
THT TTH↵
HHH TTT↵
HTT HTT↵
```

Sample Output

```
HHT↵
TTH↵
TIE↵
TIE↵
```