AUPC Practice Contest 2

2025-03-28

Problems

A. Matchsticks
B. (More) Multiplication
C. ASCII Addition
D. Bank Queue
E. A Prize No One Can Win
F. A New Alphabet

Advice, hints, and general information

- Your solution programs should read input from standard input (e.g. System.in in Java or std::cin in C++) and produce output on standard output (e.g. System.out in Java or std::cout in C++). Anything written on standard error will be ignored. For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is “No comment, read problem statement”, indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem.

Matchsticks

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Matchsticks are ideal tools to represent numbers. A common way to represent the ten decimal digits with matchsticks is the following:

This is identical to how numbers are displayed on an ordinary alarm clock. With a given number of matchsticks you can generate a wide range of numbers. We are wondering what the smallest and largest numbers are that can be created by using all your matchsticks.

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with an integer n ($2 \leq n \leq 100$): the number of matchsticks you have.

Output

Per testcase:

- One line with the smallest and largest numbers you can create, separated by a single space. Both numbers should be positive and contain no leading zeroes.

Sample Input 1

```
4
3
6
7
15
```

Sample Output 1

```
7 7
6 111
8 711
108 7111111
```

(More) Multiplication

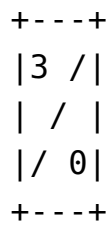
CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Educators are always coming up with new ways to teach math to students. In 2011, an educational software company, All Computer Math (ACM), developed an application to display products in a traditional grade school math format. ACM is now working on an updated version of the software that will display results in a lattice format that some students find to be easier when multiplying larger numbers.

An example would be when multiplying $345 \cdot 56 = 19320$ as given below, using a lattice grid with 2 rows and 3 columns, which appears inside a surrounding frame:

```
+---------------+
|   3   4   5   |
| +---+---+---+ |
| |1 /|2 /|2 /| |
| | / | / | / |5|
|1|/ 5|/ 0|/ 5| |
| +---+---+---+ |
|/|1 /|2 /|3 /| |
| | / | / | / |6|
|9|/ 8|/ 4|/ 0| |
| +---+---+---+ |
|/ 3 / 2 / 0    |
+---------------+
```

The first operand, 345, is displayed above the top of the grid with each digit centered horizontally above its column of the grid, and the second operand, 56, is displayed along the righthand side with each digit centered vertically at the center of its row in the grid. A single cell of the grid, such as



represents the product of the digit of the first operand that is above its column and the digit of the second operand that is to the right of its row. In our example, this cell represents the product 5 times 6 = 30 that results when multiplying the 5 in 345 and the 6 in 56. Note that the 10's digit of that product is placed in the upper left portion of this cell and the 1's digit in the lower right.

The overall product is then computed by summing along the diagonals in the lattice that represent the same place values in the result. For example, in our first problem the product 19320 was computed as:

1's digit = **0**

10's digit = 5 + 3 + 4 = 12, thus **2** with a carry of 1

100's digit = (1 carry) + 2 + 0 + 2 + 8 = 13, thus **3** with a carry of 1

1000's digit = (1 carry) + 2 + 5 + 1 = **9**

10000's digit = **1**

The resulting product is placed with the one's digit below the grid at the far right and, depending on its length, with the most significant digits wrapped around the left side of the grid. Each digit of the final product appears perfectly aligned with the corresponding diagonal summands.

To provide an aesthetic view, we use a series of minus (-) characters for horizontal lines, pipe (|) characters for vertical lines, and slash (/) characters for diagonal lines. Furthermore, we use a plus (+) character wherever a horizontal and vertical line meet. Each multiplication lattice is subsequently "boxed" by an outer border. There is a row containing the first operand which is between the topmost border and the top line of the grid, and a row between the bottom of the grid and the bottom border, which contains some portion of the resulting product. There is one column between the leading | and the left edge of the inner grid, which may contain a portion of the resulting product, and one column after the right edge of the inner grid but before the rightmost | border, which contains the second operand. If the product is not long enough to wrap around the bottom-left corner, the column between the left border and the left edge of the grid will containing only spaces. (See the later example of 3 x 3.)

Leading zeros should be displayed within lattice grid cells, but leading zeros should never be displayed in the product, nor should there ever be a slash (/) character prior to the leading digit of the product. For example, consider the product of $12 \cdot 27 = 324$ below:

```
+-----------+
|   1   2   |
| +---+---+ |
| |0 /|0 /| |
| | / | / |2|
| |/ 2|/ 4| |
| +---+---+ |
| |0 /|1 /| |
| | / | / |7|
|3|/ 7|/ 4| |
| +---+---+ |
|/ 2 / 4    |
+-----------+
```

Note that in the top-right grid of the lattice, the product $2 \cdot 2 = 04$ is displayed with the zero for the tens digit. However, there is no thousands digit displayed in the product 324, nor is there any slash displayed above the digit 3 in that product.

Input

The input is composed of T test cases ($1 \leqslant T \leqslant 20$).

Each test contains two positive integers, A and B, such that $1 \leqslant A \leqslant 9999$ and $1 \leqslant B \leqslant 9999$.

The last test case will be followed by a line containing `0 0`.

Output

For each test case, produce the grid that illustrates how to multiply the two numbers using the lattice multiplication technique.

Sample Input 1

```
345 56
12 27
1 68
9999 7
3 3
0 0
```

Sample Output 1

```
+---------------+
|   3   4   5   |
| +---+---+---+ |
| |1 /|2 /|2 /| |
| | / | / | / |5|
|1|/ 5|/ 0|/ 5| |
| +---+---+---+ |
|/|1 /|2 /|3 /| |
| | / | / | / |6|
|9|/ 8|/ 4|/ 0| |
| +---+---+---+ |
|/ 3 / 2 / 0    |
+---------------+
+-----------+
|   1   2   |
| +---+---+ |
| |0 /|0 /| |
| | / | / |2|
| |/ 2|/ 4| |
| +---+---+ |
| |0 /|1 /| |
| | / | / |7|
|3|/ 7|/ 4| |
| +---+---+ |
|/ 2 / 4    |
+-----------+
+-------+
|   1   |
| +---+ |
| |0 /| |
| | / |6|
| |/ 6| |
```

```
| +---+ |
| |0 /| |
| | / |8|
|6|/ 8| |
| +---+ |
|/ 8    |
+-------+
+--------------------+
|   9   9   9   9    |
| +---+---+---+---+ |
| |6 /|6 /|6 /|6 /| |
| | / | / | / | / |7|
|6|/ 3|/ 3|/ 3|/ 3| |
| +---+---+---+---+ |
|/ 9 / 9 / 9 / 3    |
+--------------------+
+-------+
|   3   |
| +---+ |
| |0 /| |
| | / |3|
| |/ 9| |
| +---+ |
|  9    |
+-------+
```

ASCII Addition

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Nowadays, there are smartphone applications that instantly translate text and even solve math problems if you just point your phone's camera at them. Your job is to implement a much simpler functionality reminiscent of the past – add two integers written down as ASCII art.

An ASCII art is a matrix of characters, exactly 7 rows high, with each individual character either a dot or the lowercase letter x.

An expression of the form $a + b$ is given, where both a and b are positive integers. The expression is converted into ASCII art by writing all the expression characters (the digits of a and b as well as the + sign) as 7×5 matrices, and concatenating the matrices together with a single column of dot characters between consecutive individual matrices. The exact matrices corresponding to the digits and the + sign are as folows:

```
xxxxx
x...x
x...x
x...x
x...x
x...x
xxxxx

....x
....x
....x
....x
....x
```

```
....x
....x

xxxxx
....x
....x
xxxxx
x....
x....
xxxxx

xxxxx
....x
....x
xxxxx
....x
....x
xxxxx

x...x
x...x
x...x
xxxxx
....x
....x
....x

xxxxx
x....
x....
xxxxx
....x
....x
xxxxx

xxxxx
x....
```

```
x....
xxxxx
x...x
x...x
xxxxx

xxxxx
....x
....x
....x
....x
....x
....x

xxxxx
x...x
x...x
xxxxx
x...x
x...x
xxxxx

xxxxx
x...x
x...x
xxxxx
....x
....x
xxxxx

.....
..x..
..x..
xxxxx
..x..
```

```
..x..
.....
```

Given an ASCII art for an expression of the form $a + b$, find the result of the addition and write it out in the ASCII art form.

Input

Input consists of exactly 7 lines and contains the ASCII art for an expression of the form $a + b$, where both a and b are positive integers consisting of at most 9 decimal digits and written without leading zeros.

Output

Output 7 lines containing ASCII art corresponding to the result of the addition, without leading zeros.

Sample Input 1

```
....x.xxxxx.xxxxx.x...x.xxxxx.xxxx
x.xxxxx.......xxxxx.xxxxx.xxxxx
....x.....x.....x.x...x.x.....x...
......x...x...x...x.x...x.x...x
....x.....x.....x.x...x.x.....x...
......x...x...x...x.x...x.x...x
....x.xxxxx.xxxxx.xxxxx.xxxxx.xxxx
x.....x.xxxxx.xxxxx.xxxxx.x...x
....x.x.........x.....x.....x.x...
x.....x...x...x...x.....x.x...x
....x.x.........x.....x.....x.x...
x.....x...x...x...x.....x.x...x
....x.xxxxx.xxxxx.....x.xxxxx.xxxx
x.....x.......xxxxx.xxxxx.xxxxx
```

Sample Output 1

```
....x.xxxxx.xxxxx.xxxxx.x...x.xxxx
x.xxxxx
....x.....x.....x.x.....x...x.x...
......x
....x.....x.....x.x.....x...x.x...
......x
....x.xxxxx.xxxxx.xxxxx.xxxxx.xxxx
x.....x
....x.x.........x.....x.....x.....
x.....x
....x.x.........x.....x.....x.....
x.....x
....x.xxxxx.xxxxx.xxxxx.....x.xxxx
x.....x
```

Bank Queue

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Oliver is a manager of a bank near KTH and wants to close soon. There are many people standing in the queue wanting to put cash into their accounts after they heard that the bank increased the interest rates by 42% (from 0.01% per year to 0.0142% per year).
However, there are too many people and only one counter is open which can serve one person per minute. Greedy as Oliver is, he would like to select some people in the queue, so that the total amount of cash stored by these people is as big as possible and that money then can work for the bank overnight.

There is a problem, though. Some people don't have the time to wait until the bank closes because they have to run somewhere else, so they have to be served before a certain time, after which they just leave. Oliver also turned off the infrared door sensor outside the bank, so that no more people can enter, because it's already too crowded in the hall.

Task

Help Oliver calculate how much cash he can get from the people currently standing in the queue before the bank closes by serving at most one person per minute.

Input

The first line of input contains two integers N ($1 \leq N \leq 10\,000$) and T ($1 \leq T \leq 47$), the number of people in the queue and the time in minutes until Oliver closes the bank. Then follow N lines, each with 2 integers c_i and t_i, denoting the amount of cash in Swedish crowns person i has and the time in minutes from now after which person i leaves if not served. Note

that it takes one minute to serve a person and you must begin serving a person at time t_i at the latest. You can assume that $1 \leq c_i \leq 100\,000$ and $0 \leq t_i < T$.

Output

Output one line with the maximum amount of money you can get from the people in the queue before the bank closes.

Sample Input 1

```
4 4
1000 1
2000 2
500 2
1200 0
```

Sample Output 1

```
4200
```

Sample Input 2

```
3 4
1000 0
2000 1
500 1
```

Sample Output 2

```
3000
```

A Prize No One Can Win

CPU TIME LIMIT	MEMORY LIMIT
2 seconds	1024 MB

Image from https://www.maxpixel.net/Girl-Young-Cute-Mammal-Unicorn-Animal-Portrait-3187993.

After the festive opening of your new store, the Boutique store for Alternative Paramedicine and Cwakhsahlvereigh, to your disappointment you find out that you are not making as many sales as you had hoped. To remedy this, you decide to run a special offer: you will mark some subset of the n items for sale in your store as participating in the offer, and if people buy exactly two of these items, and the cost of these items is *strictly* more than X euros, you will give them a free complimentary unicorn horn!

Since you recently found out all your unicorn horns are really narwhal tusks, you decide to rig the offer by picking the participating items in such a way that no one can earn a horn anyway.

To make sure no one becomes suspicious, you want to mark as many items as possible as participating in the offer.

Input

- On the first line are two integers, $1 \leq n \leq 10^5$, the number of items for sale in your store, and $1 \leq X \leq 10^9$, the minimum cost specified in the statement.
- On the second line are n positive integers, each at most 10^9. These are the prices of the items in the store.

Output

Print the maximum number of items you can mark as part of your special offer, without anyone actually being able to receive a horn.

Sample Input 1

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

Sample Output 1

```
3
```

Sample Input 2

```
5 10
4 8 1 9 7
```

Sample Output 2

```
2
```

Sample Input 3

```
4 10
1 3 1 7
```

Sample Output 3

```
4
```

Sample Input 4

```
1 5
6
```

Sample Output 4

```
1
```

A New Alphabet

CPU TIME LIMIT	MEMORY LIMIT
1 second	1024 MB

Photo by r. nial bradshaw

A New Alphabet has been developed for Internet communications. While the glyphs of the new alphabet don't necessarily improve communications in any meaningful way, they certainly make us *feel cooler*.

You are tasked with creating a translation program to speed up the switch to our more *elite* New Alphabet by automatically translating ASCII plaintext symbols to our new symbol set.

The new alphabet is a one-to-many translation (one character of the English alphabet translates to anywhere between 1 and 6 other characters), with each character translation as follows:

Original	New	English Description	Original	New	English Description
a	@	at symbol	n	[]\[]	brackets, backslash, brackets
b	8	digit eight	o	0	digit zero
c	(	open parenthesis	p	|D	bar, capital D
d	|)	bar, close parenthesis	q	(,)	parenthesis, comma, parenthesis
e	3	digit three	r	|Z	bar, capital Z
f	#	number sign (hash)	s	$	dollar sign
g	6	digit six	t	']['	quote, brackets, quote
h	[-]	bracket, hyphen, bracket	u	|_|	bar, underscore, bar
i	|	bar	v	\/	backslash, forward slash
j	_|	underscore, bar	w	\/\/	four slashes
k	|<	bar, less than	x	}{	curly braces

For instance, translating the string “Hello World!” would result in:

```
[-]3110 \/\/0|Z1|)!
```

Note that uppercase and lowercase letters are both converted, and any other characters remain the same (the exclamation point and space in this example).

Input

Input contains one line of text, terminated by a newline. The text may contain any characters in the ASCII range 32–126 (space through tilde), as well as 9 (tab). Only characters listed in the above table (A–Z, a–z) should be translated; any non-alphabet characters should be printed (and not modified). Input has at most $10\,000$ characters.

Output

Output the input text with each letter (lowercase and uppercase) translated into its New Alphabet counterpart.

Sample Input 1

```
All your base are belong to us.
```

Sample Output 1

```
@11 `/0|_||Z 8@$3 @|Z3 8310[]\[]6
'][']0 |_|$.
```

Sample Input 2

```
What's the Frequency, Kenneth?
```

Sample Output 2

```
\/\/[-]@'][''$            ']['[-]3
#|Z3(,)|_|3[]\[](`/,   |<3[]\[][]\
[]3']['[-]?
```

Sample Input 3

```
A new alphabet!
```

Sample Output 3

```
@ []\[]3\/\/ @1|D[-]@83'][''!
```