Assignment 2

Rules[1]:

1. There are 5 problems to complete.
2. All questions require you to read the test data from a input file (“input.txt”) and write the output in an output file (“output.txt”). Follow the input/output format for each problem.
3. The allowed programming languages are C, C++, Java and Python.
4. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available.
5. Each problem should be finished in less than 1 Minute (If you come up with a good algorithm it should not take even more than 5 sec. Anyways you can try your code on machines in D242).
6. The input to all problems will consist of multiple test cases. (Try your code with as many test cases as you can).
7. For each problem submit just one file with the name of the problem (for example for problem A, your file should be A.cpp or A.py ,...)
8. Put all your source codes in a folder, compress it then submit it to the blackboard.

[1] The header is just added for fun and the sake of copyright (the problems are chosen form previous ICPC contests). Of course this is not a contest! ;)

acm International Collegiate Programming Contest

Problem A

Before bridges were common, ferries were used to transport cars across rivers. River ferries, unlike their larger cousins, run on a guideline and are powered by the river's current. Two lanes of cars drive onto the ferry from one end, the ferry crosses the river, and the cars exit from the other end of the ferry.

The cars waiting to board the ferry form a single queue, and the operator directs each car in turn to drive onto the port (left) or starboard (right) lane of the ferry so as to balance the load. Each car in the queue has a different length, which the operator estimates by inspecting the queue. Based on this inspection, the operator decides which side of the ferry each car should board, and boards as many cars as possible from the queue, subject to the length limit of the ferry. Your job is to write a program that will tell the operator which car to load on which side so as to maximize the number of cars loaded.

Input

There will be several test cases in the input file (“input.txt”). Each test case will begin with a line with two integers ***n*** ($1 \le n \le 100$) and ***m*** ($1 \le m \le 1000$), where ***n*** is the length of the ferry (in metres), ***m*** is the number of cars. For each car in the queue there is a line in the input specifying the length of the car (in cm, an integer between 100 and 3000 inclusive). The cars must be loaded in order, subject to the constraint that the total length of cars on either side does not exceed the length of the ferry. Subject to this constraint as many cars should be loaded as possible, starting with the first car in the queue and loading cars in order until no more can be loaded. The input will end with a line with two ***0***s

Output

Write the output in a file (“output.txt”). For each test case, print out the names of the cars, that can be loaded onto the ferry. For each car that can be loaded onto the ferry, in the order the cars appear in the input, output a line containing "port" if the car is to be directed to the port side and "starboard" if the car is to be directed to the starboard side. If several arrangements of the cars meet the criteria above, any one will do. Do not output any spaces. Do not separate answers with a blank line.

Sample Input

```
50 7
2500
3000
1000
1000
1500
700
800
0 0
```

Sample Output

```
6
port
starboard
starboard
starboard
port
port
```

Problem B

Soundex coding groups together words that appear to sound alike based on their spelling. For example, "can" and "khawn", "con" and "gone" would be equivalent under Soundex coding.

Soundex coding involves translating each word into a series of digits in which each digit represents a letter:

1 represents B, F, P, or V
2 represents C, G, J, K, Q, S, X, or Z
3 represents D or T
4 represents L
5 represents M or N
6 represents R

The letters A, E, I, O, U, H, W, and Y are not represented in Soundex coding, and repeated letters with the same code digit are represented by a single instance of that digit. Words with the same Soundex coding are considered equivalent.

Input

There will be several test cases in the input file (“input.txt”). Each test case will consist of a single word all uppercase, less than 20 letters long.

Output

Write the output in a file (“output.txt”). For each test case, For each line of input, produce a line of output giving the Soundex code. Output no spaces, and do not separate answers with blank lines.

Sample Input	Sample Output
KHAWN	25
PFISTER	1236
BOBBY	11

Problem C

Old gas stations used plastic digits on an illuminated sign to indicate prices. When there is an insufficient quantity of a particular digit, the attendant may substitute another one upside down.

The digit "6" looks much like "9" upside down. The digits "0", "1" and "8" look like themselves. The digit "2" looks a bit like a "5" upside down (well, at least enough so that gas stations use it).

Due to rapidly increasing prices, a certain gas station has used all of its available digits to display the current price. Fortunately, this shortage of digits need not prevent the attendant from raising prices. She can simply rearrange the digits, possibly reversing some of them as described above.

Your job is to compute, given the current price of gas, the next highest price that can be displayed using exactly the same digits.

Input

There will be several test cases in the input file (“input.txt”). The input consists of several lines, each containing between 2 and 30 digits (to account for future prices) and a decimal point immediately before the last digit. There are no useless leading zeros; that is, there is a leading zero only if the price is less than 1. The input will end with a line with a decimal point alone.

Output

Write the output in a file (“output.txt”). You are to compute the next highest price that can be displayed using the same digits and the same format rules. If the price cannot be raised, print the original price. Output no extra spaces, and do not separate answers with blank lines.

Sample Input	Sample Output
65.2	65.5
76.7	77.6
77.7	77.7
.	

Problem D

Wine bottles are never completely filled: a small amount of air must be left in the neck to allow for thermal expansion and contraction. If too little air is left in the bottle, the wine may expand and expel the cork; if too much air is left in the bottle, the wine may spoil. Thus each bottle has a minimum and maximum capacity.

Given a certain amount of wine and a selection of bottles of various sizes, determine which bottles to use so that each is filled to between its minimum and maximum capacity and so that as much wine as possible is bottled.

Input

There will be several test cases in the input file (“input.txt”). The first line of each test case will consist of two integers: the amount of wine to be bottled (in litres, between 0 and 1,000,000) and the number of sizes of bottles (between 1 and 100). For each size of bottle, one line of input follows giving the minimum and maximum capacity of each bottle in millilitres. The maximum capacity is not less than 325 ml and does not exceed 4500 ml. The minimum capacity is not less than 95% and not greater than 99% of the maximum capacity. You may assume that an unlimited number of each bottle is available. The input will end with a line with two 0s.

Output

Write the output in a file (“output.txt”). For each test case, output a single integer on its own line, indicating the amount of wine, in ml, that cannot be bottled.
Output no spaces, and do not separate answers with blank lines.

Sample Input	Sample Output

10 2
4450 4500
725 750
	250

10000 2
4450 4500
725 750
	0

0 0	

Problem E

A (formal) language is a set of strings. One way to define a particular language is using ordinary set notation. Alternatively, some form of grammar may be more convenient for representing large sets. The magic grammar in which we are interested has two parts:

- An initial string
- A set of replacement rules of the form s_1 -> s_2 where s_1 and s_2 are strings

The language defined by this grammar is the set of all strings that can be generated by repeatedly replacing s_1 by s_2 within the initial string.

For example, consider the grammar G consisting of the initial string
AyB

and the replacement rules
{A->ab , Ay->cdy , B->w , B->x} .

G generates the language
L = {AyB, Ayw, Ayx, abyB, abyw, abyx, cdyB, cdyw, cdyx}

Given a magic grammar G, compute how many different strings there are in the language generated by G.

Input

There will be several test cases in the input file ("input.txt"). The first line of each test case will consist of a string which the initial string and an integer n (0<n<=100) which is the number of replacement rules. The subsequent n lines contain the replacement rules, one per line. Each rule contains two strings separated by space. Each string contains between 0 and 10 upper and lower case letters. The input will end with line with two 0s.

Output

Write the output in a file ("output.txt"). For each test case, output a line with the number of distinct strings in the language generated by G. If there are more than 1000 distinct strings, print "Too many." instead.

Sample Input	Sample Output
AyB 4 A ab Ay cdy B w B x	9
0 0	