United Kingdom and Ireland Subregional Contest 2016

29th October 2016

ACM International Collegiate Programming Contest

IBM event sponsor

Problems

A Taxing Problem
B Build a Boat
C Compiler
D The Darkness
E Elegant Showroom
F Fridge
G Gondolas
H Rhyming Slang
I Grass Seed Inc.
J Jack and the Beanbag
K Compensation
L Secret Santa

Problems are not ordered by difficulty.
Do not open before the contest has started.

This page is intentionally left (almost) blank.

Problem A
Taxing Problem

George has won the lottery and, being a nice guy, has decided to spread the wealth around. However, monetary gifts can be taxed once they get over a certain size—the amount of tax depends on how much his friends have earned that year.

The amount of tax paid depends on tax bands. The bands start at zero. Each one takes a certain cut of income from the range of pre-tax income it covers. The final tax band applies to all income above its lower bound.

George is a savvy fellow and knows the number of tax bands, the amount of money each friend has earned and the amount he wants each friend to walk away with.

How much should George give to each friend before tax?

Input

- One line containing an integer B ($1 \leq B \leq 20$): the number of tax bands.
- B further lines, each containing two real numbers: s_i ($0 < s_i \leq 10^6$): the size in pounds of the i^{th} tax band, and p_i ($0 \leq p_i \leq 100$): the percentage taxation for that band.
- One line containing a real number P ($0 \leq P \leq 99.999$): the percentage tax on all income above other bands.
- One line containing an integer F, ($0 < F \leq 20$): the number of friends George wants to pay.
- F further lines, each containing two real numbers e_j and m_j ($0 \leq e_j \leq 10^6$, $0 < m_j \leq 10^6$): the amount of money the j^{th} friend has earned, and the amount of money they should receive after tax respectively.

Tax bands will be given in increasing order.

Output

- F lines, one for each friend specified in the input and in the same order.

  Each line should contain one real number: the amount of money George will give to his friend, in order to arrive at the correct amount after they have paid tax.

All output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1

```
1
1000 0
20
3
0.0 500
999.5 500
1000.0 500
```

Sample Output 1

```
500.000000
624.875000
625.000000
```

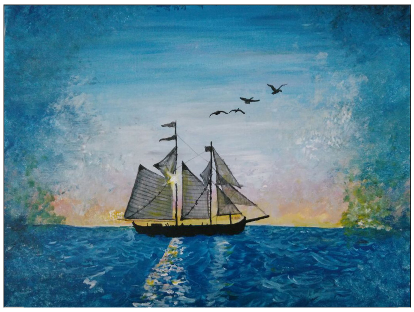
Sample Input 2

```
3
4750.50 0
8000 20
10000 40
60
3
0 10000
10000 5000
15000 5000
```

Sample Output 2

```
11312.375000
7416.500000
8624.750000
```

Problem B
Build a Boat



An oft-forgotten part of a well-rounded software engineer's training is those long but vital months spent learning the art of shipwrighting.

Modern boats, as we know, are superbly safe, to the point that they are nigh unsinkable. Even in a head-on collision the ship can be saved by a system of *bulkheads*, reinforced vertical sections inside the structure designed to prevent ingressed water from spreading to other sections.

A splendid new ship of the line has had a team of talented artists hard at work reticulating the finest splines for use in this vessel. However, not being concerned with such details, they have left out the placements of the bulkheads as an exercise for their readers.

This is where you can help. First, we need to find how many bulkheads we can fit in the ship. Second, the exact placements of the bulkheads need to be found.

Input

- One line containing an integer C ($10 \leq C \leq 10^9$), the minimum area of a bulkhead section.
- One line containing an integer N ($3 \leq N \leq 10^5$), the number of vertices in the artists' design for the boat.
- N lines, each containing two positive integers: x and y ($-10^4 \leq x, y \leq 10^4$), the coordinates of the vertices from the hull in counter-clockwise winding order.

The shape of the boat never doubles back on itself horizontally; that is to say, if a vertical line is drawn through the cross-section, no matter where, it will always pass through the boat exactly once—never twice.

It is guaranteed that it is always possible to fit at least one bulkhead section into the ship.

Output

- One line containing one integer, M: the maximum number of bulkhead sections that can be created. It is guaranteed that M is between 1 and 100.
- $M - 1$ lines, each containing one real number: the X-coordinate of the placement of a bulkhead such that the sections defined by it have equal area to all the others. Bulkhead placements must be given in increasing order of X.

All output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
50 4 110 10 80 10 80 0 110 0	6 85 90 95 100 105

Sample Input 2	Sample Output 2
24 3 10 10 30 10 20 20	4 17.071067 20 22.928932

Sample Input 3	Sample Output 3
1280 10 100 120 97 50 94 99 74 97 50 87 29 71 13 50 3 26 0 0 100 0	6 27.5015466 44.3204382 59.0041321 72.7008423 85.8494453

Problem C
Compiler

One thing almost all computers have in common, whether it is a simple stack-based calculator, a 6502 powered BBC Micro, or a brand new Android phone, every modern computer requires programs that turn either high-level languages or assembly language into machine code.

UKIEPC recently designed their own processor. While very easy to load programs onto, it is not as complex as some, since it has only one aim: show a number on an innovative laser display board!

The available memory consists of three 8-bit registers: `A`, `X`, and `Y`, plus an infinite stack. At program start these registers are initialised to unknown values, and the stack is empty.

The processor supports six unique instructions:

- `PH <reg>`: push the contents of the register (i.e. `A`, `X`, or `Y`) onto the stack.
- `PL <reg>`: pop a value off the stack into the register. The program will terminate if this instruction is called when the stack is empty.
- `AD`: pop two values off the stack, and push the lowest 8 bits of their sum onto the stack.
- `ZE <reg>`: set the register equal to zero.
- `ST <reg>`: set the register equal to one.
- `DI <reg>`: send the value of the register to the laser display board and exit.

Due to memory constraints, the maximum number of instructions that can be written to disk is 40. Further instructions will not be executed.

Given a number, write a program to output the number on the laser display board.

Input

- One line containing an integer N ($0 \leq N \leq 255$), the number to output.

Output

- At most 40 lines, each containing one valid instruction.

  When run in sequence the lines should output the number N. The last instruction should be a `DI`.

Sample Input 1	Sample Output 1
2	ST A ST X PH A PH X AD PL Y DI Y

Sample Input 2	Sample Output 2
5	ST X PH X PH X PH X AD PL Y PH Y PH Y AD AD PL A DI A

Problem D
The Darkness

Night clubs aren't what they used to be. Our benevolent state has decided that, for health and safety reasons, every club must now meet a minimum lighting standard. They have moved to verify standards by dividing each club up into $1m^2$ cells and measuring the light levels on the floor with probes in the centre of each cell.

Club owners aren't happy about this; they will comply with the letter of the law, but don't want to spend a penny more than they have to on electricity. Instead of upgrading their lighting, they will fence off the offending parts of the club to deal with later.

You will be given a grid representing a map of the lights in a club. Each cell of the grid (r, c) will have a light directly above its centre with a bulb of strength s, $(0 \leq s \leq 9)$.

The ceiling is flat—its height is constant. Each light will affect every nearby cell, increasing the light level at distance (x, y, z) by:

$$\frac{s}{x^2 + y^2 + z^2}$$

Building a section of transparent fencing between any two cells usually costs £11. However, if the cells on both sides meet the minimum lighting standard, the price per length of fencing rises steeply to £43, as builders find it hard to work under the pulsating night club lights and demand extra compensation.

How much will you have to spend on fencing out the dark spots?

Input

- One line containing an integer B $(0 < B \leq 9)$, the minimum required light level.
- One line containing an integer H $(0 < H \leq 5)$, the height of the ceiling in the club.
- One line containing two integers R and C $(0 < R, C \leq 30)$, the length and width of the club.
- R further lines, each containing a string of C digits, representing the strength of lights at each cell of the club.

It is guaranteed that the $2(R + C) - 4$ cells along the borders of the club are sufficiently lit.

Output

- One line containing one integer: the total number of pounds that must be spent on fencing.

Sample Input 1

```
9
1
6 6
333333
300003
300003
300003
300003
333333
```

Sample Output 1

```
176
```

Sample Input 2

```
5
2
6 7
6323226
3000005
2000002
2000002
5000003
6223236
```

Sample Output 2

```
66
```

Problem E
Elegant Showroom

A car showroom is one of the few places where cars can be found indoors. Showrooms often have many cars, even above ground level! As cars are sold and new cars are bought in to sell, the cars must be moved carefully out of the showroom.

Clearly employees only wish to move cars if they have to. So, given a map of a showroom including its walls, doors and where the cars are, and the co-ordinates of the car to move, how many cars must be moved?

Cars can be rotated on the spot, but can only be moved through a completely empty space and not diagonally. Doors are always wide enough to move a car through.

Input

- One line containing two integers R, C ($3 \le R, C \le 400$), the size of the showroom in rows and columns.
- Another R lines, each containing a string of C characters with the following meaning:
  - '`#`': a wall;
  - '`c`': a car;
  - '`D`': a door in a wall.

  The first and last lines must be walls or doors. The first and last characters in a row must be walls or doors.
- The next line will contain two integers r ($1 < r < R$), and c ($1 < c < C$), the co-ordinates of the car to move. $1, 1$ is the top-left corner.

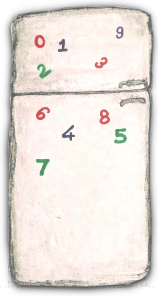
Output

- One line containing one integer: the smallest number of cars that need to be moved (including the car we are moving) to allow our desired car to leave the building.

Sample Input 1	Sample Output 1
4 5 ##### #cDc# #c#cD ##### 3 2	4

Sample Input 2	Sample Output 2
10 10 ########## #cc#ccccc# #cc#cccccD #cccccccc# ########c# #cccccccc# ###cccccc# #c#cccccc# #cccccccc# ########## 2 2	11

Problem F
Fridge



The technology behind the fridge has changed little over the years. Even so, many of the original owners of the Fred W. Wolf domestic refrigerator of 1913 would be amazed by the size and features of the modern appliances. However, since the 1960s one thing has been common for all fridge owners around the world: fridge magnets.

An effective, albeit lazy, way to keep a small child entertained is to supply them with a set of magnetic numbers and a large magnetic surface, such as said fridge, to provide the playing field upon which to apply these digits.

Far from a time-wasting exercise, this provides valuable training in the mathematical field of *counting*: moving the digits around to form "1", "2", and so on up to such heights as "10", "11", "12", and even beyond.

The possibilities are endless! ...Or at least, they would be, if the supply of digits was not limited. Given the full list of what numbers we are in possession of, what is the smallest positive number that *cannot* be made using each of digits at most once?

Input

- One string of at most 1000 digits, containing the available digits in no particular order.

Output

- One line containing one positive integer: the smallest natural number that it is not possible to assemble from the supplied digits.

Sample Input 1	Sample Output 1
`7129045863`	`11`

Sample Input 2	Sample Output 2
`55`	`1`

Sample Input 3	Sample Output 3
`123456789`	`10`

Problem G
Gondolas

The most adventurous part of skiing is the journey onto the mountain-top, between trees and through clouds, and past all sorts of enchanting views.

Naturally, the skiers at the foot of the lift can hardly wait to take their turns (although they are a little disappointed that the climb will eventually terminate). They all know exactly which times they plan to catch a lift on the tireless rotary machine.

Unfortunately, there are only so many gondolas available at the start of the day to attach to the track loop. The track loop takes $2 \cdot T$ minutes to cycle around (T on the way up and then T on the way back down). Given that you can arrange the gondolas on the track however you see fit, what is the minimum summed waiting time of all skiers that you can achieve?

Input

- One line containing three integers:
  - N ($1 \leq N \leq 400$), the number of skiers.
  - T ($1 \leq T \leq 720$), the time to travel from the bottom of the hill to the top in minutes.
  - G ($1 \leq G \leq 400$), the number of available gondola cabs.
- A further N lines in no particular order, each containing one integer X ($0 \leq X \leq 10^6$) which gives the time when a skier arrives at the foot of the mountain.

Output

- One line containing one integer: the minimum possible sum of all waiting times, where the waiting time for one skier is the time difference between their arrival and departure on the next available gondola (which may be shared with any number of other skiers).

Sample Input 1	Sample Output 1
4 10 2 0 15 30 45	10

Sample Input 2	Sample Output 2
4 10 3 0 15 30 45	5

Sample Input 3	Sample Output 3
5 16 3 16 7 5 8 1	4

Problem H
Rhyming Slang

Rhyming slang involves replacing a common word with a phrase of two or three words, the last of which rhymes with the original word. For example,

- replacing the word "stairs" with the rhyming phrase "apples and pears",
- or replacing "rotten" with the phrase "bales of cotton".

English has such a wide variety of spellings and pronunciations that for any non-native speaker telling what rhymes isn't always easy. Perhaps you can help?

Typically, two words rhyme (or can be forced to rhyme) if both of their endings can be found on the same list of word endings that sound the same.

Given a common word, a number of lists, each containing word endings that sound the same, and a number of phrases, determine if those phrases could be rhyming slang.

Input

- One line containing the single common word S ($1 \leq |S| \leq 20$).
- One line containing an integer E ($1 \leq E \leq 10$), the number of lists of word endings that sound the same.
- E lines, each no more than 100 characters long. Each a list of space-separated word endings.
- One line containing an integer P ($1 \leq P \leq 10$), the number of phrases to test.
- P lines, each no more than 100 characters long, containing a phrase p_i of two or three words that might rhyme with the common word.

All words and letters will be in lower case. The common word's ending will appear in at least one ending list.

Output

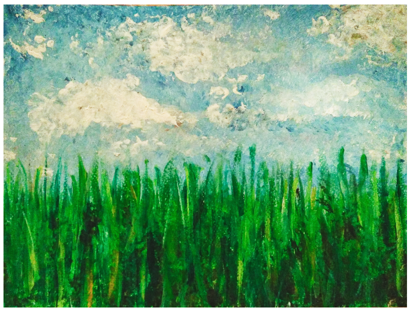
- P lines, each consisting of either:
  - 'YES': The phrase p_i rhymes with the common word.
  - 'NO': The phrase p_i does not rhyme with the common word.

Sample Input 1	Sample Output 1
`stairs` `2` `erres airs ears ares aires` `eet eat` `2` `apples and pears` `plates of meat`	`YES` `NO`

Sample Input 2	Sample Output 2
`drought` `2` `aught ought aut acht` `ought oubt outte out oute` `5` `tasty sprout` `difficult route` `worried and fraught` `forever in doubt` `apples and pears`	`YES` `YES` `YES` `YES` `NO`

Sample Input 3	Sample Output 3
`listen` `1` `issen isten ison` `2` `glisten` `listen`	`YES` `YES`

Problem I
Grass Seed Inc.



Many years ago after another unfruitful day in Cubicle Land, banging her head against yet another cutting edge, marketing buzzword-filled JavaScript framework, Janice the engineer looked out of the window and decided that time was ripe for a change.

So swapping her keyboard and mouse for a fork and a spade, she started her own gardening company.

After years of hard outdoor work Janice now has biceps like Van Damme and owns the premiere landscaping company in the whole of the South West, and has just been lucky enough to broker a large contract to sow lawns for landed gentry.

Each contract details the size of the lawns that need to be seeded, and the cost of seed per square metre. How much do you need to spend on seed?

Input

- One line containing a floating point number C ($0 < C \leq 100$), the cost of seed to sow one square metre of lawn.
- One line containing an integer L ($0 < L \leq 100$), the number of lawns to sow.
- L lines, each containing two positive floating point numbers: w_i ($0 \leq w_i \leq 100$), the width of the lawn, and l_i ($0 \leq l_i \leq 100$), the length of the lawn.

Output

- One line containing a real number: the cost to sow all of the lawns.

All output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
``2 3 2 3 4 5 5 6```	`112.0000000`

Sample Input 1

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

Sample Output 1

```
112.0000000
```

Sample Input 2

```
0.75
2
2 3.333
3.41 4.567
```

Sample Output 2

```
16.6796025
```

Problem J
Jack and the Beanbag

Jack, naïve fellow that he is, has fallen into the clutches of a dastardly and sophisticated multi-level marketing scheme.

It all started when a mysterious stranger pushed upon young Jack a bag of ordinary beans, promising that if only he could amass the right quantities of each kind of bean, he could grow a mighty beanstalk and climb it to the unimaginable wealth at its top.

This all sounds very sensible to Jack... But there is one catch. He must acquire the extra beans from other farmers, who as one might expect are not too keen to give away the fruits (nor the seeds) of their labour. Each time Jack comes to ask for a bean, they will give him exactly one from their farm, but since he is not a paying customer the exact species may vary between farmers and between visits.

There is another option, but it is expensive. Jack can give up some of his cows to the mysterious stranger in exchange for one additional bean per cow. Of course, this is a drastic measure. We would like to help Jack keep as many of his cows as possible, while still achieving his goals.

How many cows will Jack need to budget for to have 100% certainty of success?

Input

- One line containing an integer B, ($1 \leq B \leq 20$), the number of types of beans available.
- One line containing B integers, $V_1 \ldots V_B$, ($0 \leq V_1 \ldots V_B \leq 100$), the number of each kind of bean required.
- One line containing T ($1 \leq T \leq 100$), the number of other farms in Jack's small village.
- T more lines, each beginning with an integer M ($1 \leq M \leq B$) giving the number of kinds of bean this farmer grows. This is followed by M more distinct integers $T_1 \ldots T_M$ ($1 \leq T_1 \ldots T_M \leq B$), each corresponding to one kind of bean.

Output

- One line containing one integer: the number of cows Jack must bring with him in order to be 100% sure of ending the day with enough beans to grow a magical beanstalk.

Sample Input 1

```
1
5
1
1 1
```

Sample Output 1

```
0
```

Sample Input 2

```
3
5 5 5
2
2 1 2
2 2 3
```

Sample Output 2

```
10
```

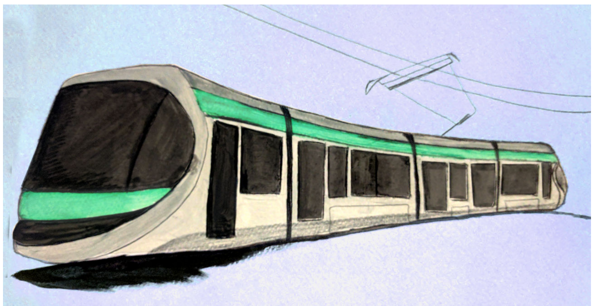
Sample Input 3

```
10
6 0 5 0 0 0 0 8 0 7
4
3 1 3 8
3 1 3 10
3 8 10 3
3 10 8 1
```

Sample Output 3

```
15
```

Problem K
Compensation



In the free-market, ruthlessly capitalist world of train fares, only one thing matters: *incentives*.

Train companies are incentivised with bonuses for high throughput, successful journeys, and customer satisfaction. Conversely, the companies are disincentivised from failure via mandatory refunds for customers delayed by 30 minutes or more.

Being a ruthless capitalist yourself, you have decided to take advantage of this generous delay compensation provision.

The refund is awarded provided that no matter the combination of trains you had taken (provided they followed the same route of stations as planned), you would still be unable to reach your destination in strictly less time than 30 minutes (or 1800 seconds), of the time you would have arrived assuming your booked journey was exactly on time.

Armed with your printout of the day's delays, and the original timetable, you must ask yourself only one question: what is the earliest time you can book a train for from station 1, in order to earn this restitutive reward?

Input

- One line containing two integers: N ($1 \le N \le 100$), the number of stations, and M ($1 \le M \le 10^5$), the number of scheduled trains.
- The next M lines each contain 4 integers:
  - X, the starting station ($1 \le X \le N - 1$),
  - S and T ($0 \le S \le T < 86400$), the planned departure and arrival times in seconds,
  - and L ($0 \le L < 86400$), the duration by which this train is delayed.

Stations are numbered from 1 to N in the order you will visit them. Each train goes between stations X and $X + 1$. It is possible to change between trains instantanesouly.

Output

- One line containing one integer: the start time of the earliest train journey you could book in order to earn your compensation, or `impossible` if no such journey is possible.

Sample Input 1	Sample Output 1
2 3 1 1800 9000 1800 1 2000 9200 1600 1 2200 9400 1400	1800

Sample Input 2	Sample Output 2
2 2 1 1800 3600 1800 1 1900 3600 1600	impossible

Sample Input 3	Sample Output 3
3 2 1 10 20 1 2 20 30 0	10

Problem L
Secret Santa

Christmas comes sooner every year. In fact, in one oft-forgotten corner of the world, gift-giving has already started in the form of a *Secret Santa* syndicate.

Everybody in the small town of Haircombe is going to put their name into a hat. This hat will be given a hearty shuffle, and then afterwards everybody will take turns once more in taking a name back from the hat.

The name each person receives is the name of the fellow citizen to whom they will send a gift.

Of course, one concern with this strategy is that some unfortunate citizens could wind up giving gifts to themselves. What are the chances that this will happen to any of the citizens of Haircombe?

Input

- One line containing the number N ($1 \leq N \leq 10^{12}$), the number of citizens who will take part in Secret Santa.

Output

- One line containing one real number; the probability that one or more people wind up giving gifts to themselves.

All output must be accurate to an absolute or relative error of at most 10^{-6}.

Sample Input 1	Sample Output 1
2	0.50000000

Sample Input 2	Sample Output 2
3	0.66666667

Sample Input 3	Sample Output 3
6	0.63194444