15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem A — Purple Balloon
Isle of the birds

Problem Description

A super-intelligent species of coconut-eating tropical bird has decided to develop a scenic flight path roughly following the perimeter of the island that they inhabit. Owing to the impressive energy consumption of their overdeveloped brains, these birds can only afford to fly in straight lines from one coconut tree to the next.

The birds reasoned that they can approximate the shape of their island (from a bird's-eye view, if you must) by imagining a rubber band that is stretched to circumscribe the island. The band is then released so that it contracts until it makes contact with the trunks of some of the coconut trees found on the island. The resulting shape of the elastic band thus forms straight lines between the coconut trees it is in contact with (allowing efficient travel), whilst maintaining the additional property that any line with endpoints inside the shape will be contained entirely within the shape. This implies that all coconut trees will fall strictly inside the shape defined by the rubber band.

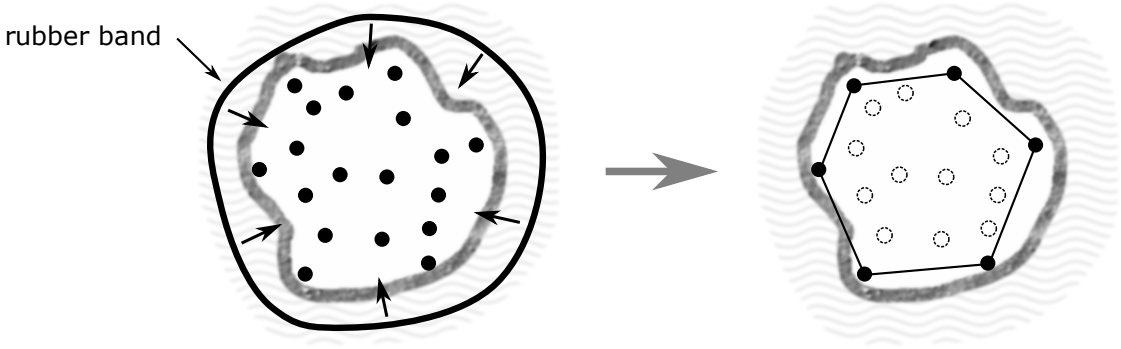

Figure 1: Illustration of how the rubber band starts out circumscribing the island, and then contracts to make contact with some of the trees.

The birds know the exact coordinates of each and every tree on the island. Using only this information, they now want to calculate the number of trees required to define the shape formed by the rubber band.

Input

Your input consists of an arbitrary number of records, but no more than 20. Each record starts with the integer value n, denoting the number of points in that record, followed by n pairs of real numbers separated by white space (one or more space and/or newline characters), with $3 \leq n \leq 15000$. Individual coordinates are in the range $[-8, 8]$ in both dimensions. Each number will have at most 20 digits after the decimal point.

The input data is guaranteed to satisfy the following property: for any three points, the triangle formed by said points will have an area of at least 10^{-10}. You may assume that the tree trunks are infinitely thin.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

```
k
```

where `k` denotes the number of trees required to define the shape formed by the hypothetical rubber band stretched around the tree trunks.

Sample input

```
5
-1.0 -1.0
1.0 1.0
-1.0 1.0
1.0 -1.0
0.1 0.3
5
1.0 0.0
0.5 -1.0
0.0 1.0
-1.0 0.0
-0.5 -1.0
-1
```

Sample output

```
4
5
```

Time limit

5 seconds

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem B — White Balloon
Street Lights

Problem Description

The small town of Acacia-Cedar Meadows (ACM) is planning to install street lights on its main road, and is trying to work out the budget for maintenance. The life time of the light bulbs follows an exponential distribution: this means that during a period of time t, an initially working light bulb fails with probability $1 - e^{-\lambda t}$, regardless of how old the bulb is (λ is a constant that is given to you). The street lights are close together, so a single broken bulb is not a problem. However, two adjacent broken bulbs will cause the lighting to be inadequate and require the town to rent a repair truck, which will then be used to replace all the broken bulbs.

Assuming that all N light bulbs are currently working, what is the expected (i.e., average) time until there are two adjacent broken bulbs?

Input

The input describes a number of test cases (up to 20). Each test case is a line containing an integer N (the number of light bulbs along the main road), and a floating-point number λ with at most four digits after the decimal point (see above for the description), separated by a space. The end of input is marked by a line containing only the value -1.

In all test cases, $2 \leq N \leq 50$ and $0.01 \leq \lambda \leq 100.0$.

Output

For each test case, print out a line containing the expected time until two adjacent bulbs have failed, with 6 digits after the decimal point. Refer to the environment manual for instructions on floating-point formatting.

Sample input

```
2 1.0
10 0.3
-1
```

Sample output

```
1.500000
1.367725
```

Time limit

1 second

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem C — Blue Balloon Fitness Training

Problem Description

John and Mary are trying to get fit. John likes to cycle, and he wants to ride A kilometres today. Mary likes to run, and she wants to run B kilometres today (B is less than A). They now want to select a circular training route, which they will go around as many times as necessary to achieve their distance targets. They have found possible routes of lengths 1, 2, ..., M. They want to choose a route such that they will end up in the same place at the end of the day for a celebratory drink (see the example for details). Find the longest route they can use.

Example

Suppose $A = 105$, $B = 42$ and $M = 10$. If they use a 10-kilometre route, then John will make 10 full circuits and then finish 5 kilometres along the route, while Mary will make 4 full circuits and finish 2 kilometres along the route. Since they don't finish in the same place, this is not acceptable. However, with a 9 kilometre route, John will make 11 full circuits and finish 6 kilometres along the route, while Mary will make 4 full circuits and also finish 6 kilometres along the route. The longest valid route in this case is thus 9.

Input

The input describes a number of test cases. Each line consists of the three positive integers A, B and M, separated by single spaces. The end of input is marked by a line containing only the value -1.

There are at most 20 test cases, and in each test case $1 \leq B < A \leq 200$ and $1 \leq M \leq 200$.

Output

For each test case, print out a single line containing the length (in kilometres) of the longest valid circuit.

Sample input

```
105 42 10
103 42 10
100 50 100
-1
```

Sample output

```
9
1
50
```

Time limit

1 second

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem D — Red Balloon
Matchstick Maths

Problem Description

For reasons known only to yourself, you have decided to write a program to solve "Matchstick maths" problems. These problems consist of a simple mathematical equation that has been expressed graphically, using matchsticks to form the digits and operators (see Figure 1). The trick of such a "Matchstick maths" equation is that the expression is given with one of the matchsticks having been moved from one possible location to another possible location, thereby creating an expression that is either badly formed (i.e., some of the digits or operators may not be represented correctly), or an expression that does not evaluate to true (e.g., 1+1=3).

Your task is to check whether a given expression can be made to evaluate to true by moving a single matchstick from one possible location to another possible location.

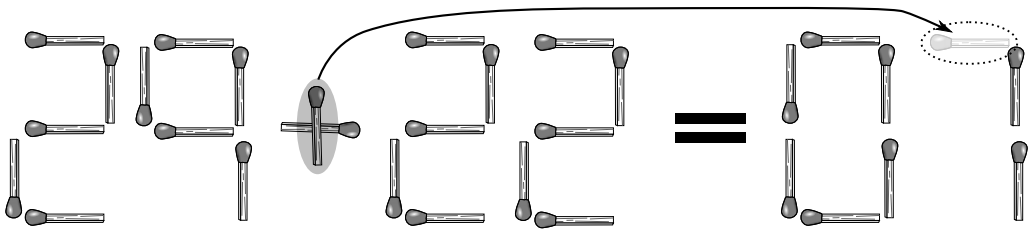

Figure 1: An example of matchstick maths: $29 + 22 = 1$ can be turned into $29 - 22 = 7$ by moving one matchstick.

Input

Your input consists of an arbitrary number of records, but no more than 20. Each record contains exactly 44 integers chosen from the set $\{0, 1\}$. Each integer represents a single location in the equation where a matchstick may be placed, with a value of "1" denoting the presence of a matchstick. Integers are separated by one or more spaces. Figure 2 shows how the possible locations are numbered, while Figure 3 provides the reference representation for each digit. Any arbitrary sequence of "0" and "1" values may appear in the input record, so take note that the input is not limited to the digits of Figure 3.

Individual numbers are limited to the range $[0, 99]$. Numbers in the range $[0, 9]$ are always represented with a leading zero, e.g., 00 for 0, or 04 for 4.

The end of input is indicated by a line containing only the value -1.

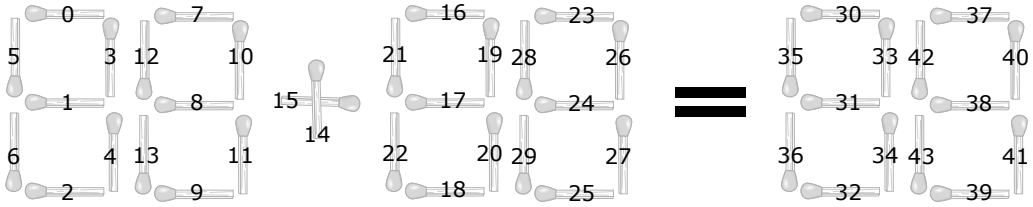

Figure 2: Numbering of match positions.

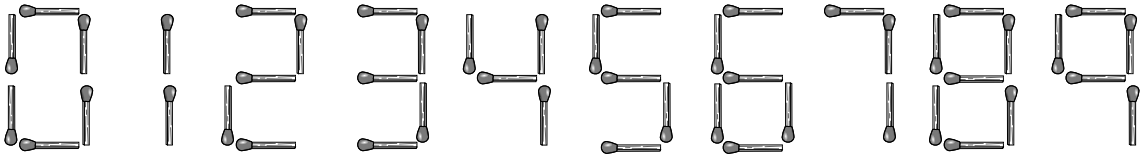

Figure 3: The representation used for each of the digits.

Output

For each input record, output either

`valid`

or

`invalid`

with `valid` being output only if a given equation can be transformed into a valid equation (left hand side of equation evaluates to the same value as the right hand side of the equation) by moving exactly one matchstick.

A valid (true) expression requires that either a plus (+) or minus (−) operator appears in positions 14 and 15 (see Figure 2), and that all the characters are well-formed (i.e., one of the characters in Figure 3).

Sample input

```
1 1 1 1 0 0 1  1 1 0 1 1 1 0  1 1  1 1 1 1 0 0 1  1 1 1 1 0 0 1  1 0 1 1 1 1 1  0 0 0 1 1 0 0
1 0 1 1 1 1 1  0 0 0 1 1 0 0  0 1  1 0 1 1 1 1 1  0 0 0 1 1 0 0  1 0 1 1 1 1 1  1 1 1 1 0 0 1
-1
```

Sample output

```
valid
invalid
```

Time limit

1 second

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem E — Orange Balloon Similarity

Problem Description

You are developing a search engine to dethrone AltaVista (ok, so maybe you travelled 15 years back in time first). The secret sauce to your search engine is the way in which you allow the users to misspell their search keywords, but still be able to find the results they were looking for.

The distance from `word_A` to `word_B` is the minimum cost of the changes that must be applied to transform `word_A` into `word_B` or to transform `word_B` into `word_A`. You allow the following changes, with their associated cost:

1. A character may be inserted into a word at a cost of 2 units, or
2. A character may be deleted from a word at a cost of 2 units, or
3. A character may be replaced with another character. If the replacement differs only in case (e.g., 'A' replaced with 'a' or vice versa), then the cost is 1 unit. For all other replacements, the cost is 2 units.

For example, consider distance from the name *pieter* to the name *peter*. We can transform *peter* into *pieter* by inserting the character 'i' after the 'p', which amounts to only one change.

If instead we measure the distance between *pieter* and *pierre*, we find that three changes are required: either we replace the last three characters of *pierre* with 'ter', or we delete the first 'r', replace the next 'r' with a 't', and insert another 'r' at the end. From these distances we can deduce that *pieter* is more similar to *peter* than it is to *pierre*.

Let $d(\texttt{keyword}, \texttt{candidate})$ denote the minimum cost of the of changes required to transform `keyword` into `candidate`, or vice versa.

The user has entered the term `keyword` in your search engine, and you have identified (by mysterious means, if necessary) a possible candidate word from your dictionary, referred to as `candidate`. Your task is to compute $d(\texttt{keyword}, \texttt{candidate})$.

Input

Your input consists of an arbitrary number of records (no more than 20), with each record containing two words separated by white space (one or more space and/or newline characters). Records are separated by a single newline character. Each word consists of n characters from the set $\{[\mathrm{a}, \mathrm{z}] \cup [\mathrm{A}, \mathrm{Z}] \cup [0, 9]\}$, with $1 \leq n \leq 10000$. The two words in each record correspond to the `keyword` and `candidate`, respectively.

The end of input is indicated by a line containing only the value -1.

Output

For each input record, output

```
k
```

where `k` denotes the minimum cost of the changes required to transform `keyword` into `candidate`.

Sample input

```
TATTACTA
TATTAAATA
polyvinyl Polynomial
-1
```

Sample output

```
4
11
```

Time limit

10 seconds

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem F — Green Balloon Railways

Problem Description

Having become fed up with the high cost of toll roads, the Association for Commuting by Monorail (ACM) have decided to construct a high-speed rail network to connect a number of cities. They have a very limited budget, so they insist that the network is a *tree*: in other words, for any pair of cities there must be exactly one route between them (possibly going via other cities).

They have asked you to design and build the rail network. You have surveyed a number of sites for rail links (each directly connecting a pair of cities) and determined the cost for each. The more expensive the network, the larger your profit, but if you are *too* greedy, the ACM will go to a different contractor. You have decided that, amongst all possible networks that are trees, you will offer to build the second-cheapest one. Now if only you could figure out how much that will cost....

Figure 1 shows the first sample case. The cheapest tree connects 1–2, 1–4 and 2–3 (cost of 18), but the second-cheapest connects 1–2, 2–4 and 2–3 for a cost of 20.

Input

The input describes a number of test cases (up to 20). Each test case starts with a line containing two integers, N and M, separated by a space. There are N cities (numbered 1 to N) and M potential rail links. This is followed by M lines, each containing three integers a_i, b_i and c_i, separated by spaces, indicating that it is possible to connect cities a_i and b_i with a rail link of cost c_i. Note that rail links are bidirectional. In each test case:

- $3 \leq N \leq 60$, $N \leq M \leq 200$, $1 \leq c_i \leq 10\,000$, $a_i \neq b_i$.
- It is possible to make at least two different trees.
- No two potential rail links will have the same cost or connect the same pair of cities. This implies that there is only one cheapest tree.

The end of input will be followed by a line containing only the number -1.

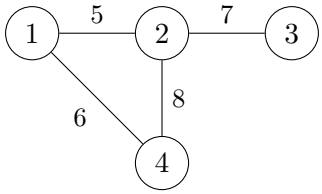

Figure 1: First sample case. Circles are cities and lines are potential rail links, with costs shown.

Output

For each test case, print out a line containing the cost of the second-cheapest tree.

Sample input

```
4 4
1 2 5
2 3 7
2 4 8
1 4 6
5 6
1 2 3
2 5 2
1 5 9
2 4 7
4 3 12
2 3 15
-1
```

Sample output

```
20
27
```

Time limit

5 seconds

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem G — Pink Balloon Necklace

Problem Description

You are trying to make a necklace from coloured beads. A necklace consists of a sequence of beads. For a necklace to be *beautiful* it must be symmetric. In other words, listing the colours of the beads from left to right must give the same list as going from right to left. Unfortunately, the store you are in does not sell individual beads. Instead, it sells packaged sets of beads which are glued together. When you buy a package of beads, you must add all the beads in the package, in the order they occur in the package, to your necklace. You cannot even turn the package around.

The store sells a number of different types of packages, and has an unlimited number of each type available. Each type of package also has a cost. You want to determine whether it is possible to make a beautiful necklace, and if so, the minimum amount it will cost.

Example

We will use upper-case letters to indicate colours. Suppose the store sells the following packages: YGYGRG for 7, BGR for 9 and GY for 3. Then you can combine YGYGRG+BGR+GY+GY to make the beautiful necklace YGYGRGBGRGYGY for a cost of 22.

Input

The input describes a number of test cases (up to 20). Each test case starts with a line containing an integer N, the number of types of packages. This is followed by N lines, each describing a package type. Each line consists of a positive integer (the cost for the package), and a string of upper-case English letters (the colours of the beads in the package).

The end of input is marked by a line containing only the value -1.

There are at most 20 packages, at most 100 beads per package, and each package costs at most 100.

Output

For each test case, print out

`C`

if a cheapest beautiful necklace will cost C, or

`IMPOSSIBLE`

if there is no way to construct a beautiful necklace.

Sample input

```
3
7 YGYGRG
9 BGR
3 GY
2
1 AB
1 CD
-1
```

Sample output

```
22
IMPOSSIBLE
```

Time limit

1 second

15^{th} South African Regional ACM Collegiate Programming Contest

Sponsored by IBM

12 October 2013

Problem H — Yellow Balloon
Student IDs

Problem Description

The brand-new University of Fourecks has accepted its first batch of students, and now needs to assign student IDs to them. The student IDs are of the form `SSSSFFNNN`, where `SSSS` are the first four consonants of the student's surname, `FF` are the first two vowels of the student's first name, and `NNN` is a three-digit sequence number used to distinguish between students who would otherwise have the same student ID (vowels are the letters A, E, I, O and U, and consonants are the other 21 letters of the English alphabet). If there are not enough consonants in the surname or vowels in the first name, fill the remaining spaces with Z.

Your task is to take the list of student names and determine the corresponding student IDs. When assigning the sequence number part of the student ID, use the smallest number (starting from 000) that will make the student ID different from those of all previous students.

Example

Suppose the students, in the order they registered, are named JOHN SMITH, MICHAEL LEE, BJORN SMITHERS, JANE SMITH and JOHN SMITH. The corresponding student IDs would be SMTHOZ000, LZZZIA000, SMTHOZ001, SMTHAE000 and SMTHOZ002. Bjorn Smithers has a sequence number of 001, because SMTHOZ000 has already been assigned to the first John Smith. That leaves the second John Smith with SMTHOZ002.

Input

The input contains the names of the students, one per line. Each line contains a first name and a surname, separated by a space. Names consist only of uppercase English letters, with no punctuation. The end of input is marked by a line containing only the string `-1`.

The input contains at most 1 000 students and the first name and surname of each student is at most 20 letters long.

Output

For each student in the input, output the corresponding student ID on a separate line. The letters must be in uppercase.

Sample input

```
JOHN SMITH
MICHAEL LEE
BJORN SMITHERS
JANE SMITH
JOHN SMITH
-1
```

Sample output

```
SMTHOZ000
LZZZIA000
SMTHOZ001
SMTHAE000
SMTHOZ002
```

Time limit

5 seconds