



Old Mutual - CSSA Computer Olympiad DAY 1



Overview

Problem	Trip	Exam	Plane
Program name	<code>trip.exe</code>	<code>exam.exe</code>	<code>plane.exe</code>
Source name	<code>trip.pas</code> <code>trip.cpp</code> <code>trip.java</code>	<code>exam.pas</code> <code>exam.cpp</code> <code>exam.java</code>	<code>plane.pas</code> <code>plane.cpp</code> <code>plane.java</code>
Input file	<code>trip.in</code>	<code>exam.in</code>	<code>plane.in</code>
Output file	<code>trip.out</code>	<code>exam.out</code>	<code>plane.out</code>
Time limit per test	5 seconds	5 seconds	5 seconds
Number of tests	10	10	10
Points per test	10	10	10
Total points	100	100	100

The maximum total score for Day1 is 300 points.



Old Mutual - CSSA Computer Olympiad Day 1



Packing for a Trip

Author: Kurt Kruger

Description

Jack (of “Jack & Jill” fame) has decided, on a whim, that he has lead too secluded a life. He has never once ventured into the city, and its bright lights and lavish lifestyles have finally caught his eye. Jill, ever the optimist, steadfastly believes that Jack will return and as such has found the smallest possible bag for Jack to pack for his trip. Jack, however, believes that after his foray into modern society, he will no longer want to return and therefore wants to pack all the necessities possible.

Task

Jack has never been a materialist and so only has a small collection of personal items, but you may assume that he has an infinite number of each item (Jill is very resourceful). Each of the K items occupies a certain volume N and has a utility value M . He wants to pack his bag so that he gains the maximum utility from the items it contains, without having the bag overflow, as his stay may be longer than just a few days.

Input (trip.in)

The input file, “trip.in”, will consist of $(2+K)$ lines of data, where the first line will consist of a single integer, C , representing the capacity of the bag. The second line consists of a single integer, K , representing the number of items Jack has to choose from. The next K lines consist of two single-space separated integers N_i and M_i which, respectively, are the volume and utility value of the i^{th} item.

NOTE: There may be duplicate lines in the input file (assume that they are simply different items occupying the same volume and having the same value).

Sample input:

trip.in

```
17
5
3 4
4 5
7 10
8 11
9 13
```

Output (trip.out)

NOTE: there are a number of correct solutions you must only output ONE.

The output file, “trip.out”, will consist of $(d+1)$ lines of data, where d is the number of items Jack has chosen. The first line must be a single integer representing the total utility value of the items Jack has placed in the bag. The following d lines must consist of two single-space separated integers representing the corresponding volume and utility value of the j^{th} item chosen.

Sample output:

trip.out

```
24
7 10
7 10
3 4
```

Constraints

The input is restricted to the following:

- 1 • C • 1024
- 1 • K • 1024
- 1 • N_i • 1024
- 1 • M_i • 2048

Time Limit:

Maximum time allowed per test case in seconds 5.

Scoring:

5 points are scored for a correct maximum utility value and another 5 points are scored if the correct items appear in the output file.

NOTE: there are a number of correct solutions you must only output ONE.



Old Mutual - CSSA Computer Olympiad Day 1



Exam

Author: Carl Hultquist

Description

Jack is very unhappy: he's stuck writing a really difficult maths exam (a course on logic, which makes things tricky for Jack since he's very illogical!) The whole exam is made up of what are termed boolean expressions, these are simply boolean variables and the three fundamental Boolean operators **and** '.', **or** '+' and **not** '!'. Here's a reminder of what these operators do:

a	b	a.b	a+b	!a	!b
F	F	F	F	T	T
F	T	F	T	T	F
T	F	F	T	F	T
T	T	T	T	F	F

Task

So we have '.' and '+' that operate on two variables, whilst '!' only operates on one variable. In the case of '.', if both variables a and b are true, then a.b is also true, otherwise a.b is false. For '+', if at least one of the variables a and b is true, then a+b is true. If neither a nor b are true, then a+b is false.

Considering '!', if a is true, then !a is false. So '!' changes the truth of the variable it applies to.

Jack might be given the following expression:

$(a+b).((b+(!c))+(!a))$

What Jack has to do is work out values for a, b and c to make the expression above true.

One set of values that works is:

a false
b true
c true

There could, however, be others, all Jack has to do is find one of these solutions. Fortunately, Jack has smuggled his cellphone into the exam and will SMS you each problem, since he knows you're good at this stuff. ;-)

Input (exam.in)

The first line of "exam.in" will consist of one integer N, $0 < N < 50$, which is the number of variables in the expression. The next N lines will each contain a string giving the name of each variable. Each variable name is limited to a single

case sensitive character. The next line of the input file will be the expression, consisting of variable names, brackets, and the three operations: '+', '.' and '!'. To avoid ambiguity, each operation will be put in its own set of brackets. The maximum length of the expression will be 255 characters.

Sample input:

exam.in

```
3
a
b
c
(a+b).((b+(!c))+(!a))
```

Output (exam.out)

The file 'exam.out' will describe exactly one satisfying solution to the given problem. 'exam.out' should consist of exactly N lines, one for each variable. Each line should consist of a single character (which is the variable being described), followed by a space, followed by either "true" or "false" to indicate whether the variable is true or false respectively. No variable may be repeated in the output, and the variable may be output in **any order**.

Sample output:

exam.out

```
a false
b true
c true
```

In this case, there are five solutions:

{ {a false, b true, c true}, {a true, b true, c true}, {a false, b true, c false}, {a true, b true, c false}, {a true, b false, c false} } any one of which could be given as the answer.

Constraints

$0 < N < 50$

$0 < \text{length of expression} \leq 255$.

Each expression is guaranteed to have a solution.

Time Limit:

Maximum time allowed per test case is 5 seconds

Scoring:

If your output file differs from the format above, you score 0.

Otherwise, you score 10 points if the solution you provide is valid (i.e. the expression is true when the variables have the values you specify).



Old Mutual - CSSA Computer Olympiad Day 1



Plane

Author: Bruce Merry

Description

Jack and Jill are worried that one day there will be a drought and they won't be able to walk up the hill to fetch water. They have come up with a drastic solution: they will create their own airline that flies all over the country, so that they can always fetch water from somewhere. However, they don't have much money to buy planes. They need your help to find how many planes they need. They also need to know where to have the planes delivered to so that they are ready for their first flights.

Task

Jack and Jill have already worked out exactly which flights they intend to offer. Unlike most airlines, flights are all one-way. Each flight starts at a particular time at one city and ends at a different city, and has a "turnaround time". The "turnaround time" is the time between departing the original city and being ready to depart the destination city. The "turnaround time" between two similar flights may differ. For example, a plane that flies an 8:00 flight from Port Elizabeth to Cape Town with a turnaround time of 3 hours can be used for flights leaving Cape Town at 11:00 or later. A similar flight from Port Elizabeth to Cape Town could have a turnaround time of 4 hours and would be ready to leave Cape Town at 12:00.

To keep things simple, the flights are the same every day. There may be multiple flights between two cities, possibly even at the same time. Due to the fact that Jack and Jill can only afford very cheap planes, flights may be overnight and may last longer than 24 hours. The planes are all the same and so any plane may be used for any flight, provided that it is at the correct city at the appropriate time. You cannot move a plane from one city to another by making an unplanned flight or by using two planes on one flight (this is why it is important to have the planes delivered to the correct cities when they are ordered).

Jack and Jill will declare the airline open at midnight on some day. All planes that are scheduled to depart at midnight will immediately depart.

Input (plane.in)

The first line of the input file "plane.in" contains two integers, N and F, separated by a single space. N is the number of cities (which are numbered 1 to N) and F is the number of flights per day.

The following F lines each describe a flight. A flight is described by 4 space separated integers: A B S T. The flight is from city A to city B, departs at S seconds after midnight (there are 86400 seconds in a day) and has a turnaround time of T seconds.

It is guaranteed that each airport has as many incoming as outgoing flights.

Sample input:

plane.in

```
4 6
1 2 0 100000
3 1 20000 10000
1 4 45000 5000
3 1 40000 10000
4 3 60000 100
2 3 10000 10000
```

Output (plane.out)

The output consists of N lines. On line i, output the number of planes that Jack and Jill must have delivered to city i.

Sample output:

plane.out

```
1
2
1
0
```

Constraints

- 2 • N • 1000
 - 1 • F • 10000
- For each flight:
- 1 • A • N
 - 1 • B • N
 - A • B
 - 0 • S < 86400
 - 0 < T < 1000000

Time Limit:

Maximum time allowed per test case is 5 seconds

Scoring:

If at any time Jack and Jill have a planned flight but no plane available to fly it, you will score 0. Otherwise you will score 100% for a correct solution, and lose 20% for each unnecessary plane that you use.