

# The Murder Mystery Contest

USC Programming Contest, Spring 2008

April 26, 2008

Sponsored by VSoE, Google, LanguageWeaver, and Electronic Arts

Once upon a time, solving murder mysteries (or, for that matter, planning murders) was the prerogative of pipe-smoking gentlemen in green checkered coats and knitting elderly women dressed in tweed. But nowadays, as we learn daily on TV, computers are irreplaceable accessories in the hunt for the culprit. After all, they are more logical than most elderly gentlemen or ladies, and tend to compute quite a bit faster. So here, we will explore some of the puzzles computers on the police force have to solve.

You can solve or submit the problems in any order you want. When you submit a problem, you submit your source code file using `PC2`. Make sure to follow the naming conventions for your source code and the input file you read. Remember that all input must be read from the file of the given name, and all output written to `stdout`. You may not use any electronic devices besides the lab computer and may not open any web browser for whatever reason.

One thing you should know about this contest is that the judging is done essentially by a `diff` of the files. That means that it is really important that your output follow the format we describe — if you have the wrong number of spaces or such, an otherwise correct solution may be judged incorrect. Consider yourselves warned!

Another warning that seems to be in place according to our experience: all of our numbering (input data sets, classes, etc.) always starts at 1, not at 0.

And a piece of advice: several problems ask you to print numbers rounded to two decimals. In C, you do this via `printf("%.2f", r);` where `r` is the variable you want to print. And in Java, the syntax is `System.out.print((new java.text.DecimalFormat("#.##")).format(r));`

[Including this page, the problem set should contain 8 pages. If yours doesn't, please contact one of the helpers immediately.]

# Problem A: A Fistful of Dollars

**File Name:** money.cpp|money.java

**Input File:** money.in

## Description

Frequently, when people kill each other, money is a primary reason<sup>1</sup>. As a result, one can often identify suspects by sudden extravagant spending. That's why the police like to have access to financial transaction records.

In this problem, you are to analyze a sequence of financial transactions, and find out if one suspect stands out as spending extravagantly. You will be given the recent purchases by a number of people. Someone is a suspect if his recent purchases add up to *more* than twice those of any other person in that period. You are to output which suspect, if any, stands out.

## Input

The first line is the number  $K$  of input data sets, followed by the  $K$  data sets, each of the following form:

The first line contains two integers  $s, t$ , the number of suspects  $2 \leq s \leq 50$  and financial transactions  $1 \leq t \leq 1000$ .

This is followed by  $t$  lines, each containing two positive integers  $s_i, p_i$ .  $s_i$  is the number of the person who made transaction  $i$  ( $1 \leq s_i \leq s$ ) and  $p_i$  the amount of money the transaction was for.

## Output

For each data set, output "Data Set  $x$ :" on a line by itself, where  $x$  is its number. On the next line, output the number of the suspect who spent more than twice as much as every other suspect in the data set. If there is no such suspect, output "No suspect." instead. Each data set should be followed by a blank line.

## Sample Input/Output

Sample input money.in

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

Corresponding output

```
Data Set 1:
No suspect.

Data Set 2:
1
```

---

<sup>1</sup>Recall the final lines from "Fargo": *So that was Mrs. Lundegaard on the floor in there. And I guess that was your accomplice in the wood chipper. And those three people in Brainerd. And for what? For a little bit of money. There's more to life than a little money, you know. Don't you know that? And here ya are, and it's a beautiful day. Well, I just don't understand it.*

# Problem B: Clue

**File Name:** `clue.cpp|clue.java`

**Input File:** `clue.in`

## Description

You may know the board game Clue, in which you put together logical clues to solve a murder mystery. You need to figure out the location, weapon, and murderer. Here are the candidates:

**Location** : Ballroom, Billiard Room, Conservatory, Dining Room, Hall, Kitchen, Library, Lounge, Study

**Weapon** : Candlestick, Gun, Knife, Lead Pipe, Rope, Wrench

**Murderer** : Colonel Mustard, Miss Scarlet, Reverend Green, Mrs. Peacock, Mrs. White, Professor Plum

Each of these locations, weapons, and suspects is printed on a card. One card of each type (location, weapon, murderer) is hidden: those are the actual ones you want to figure out. The remainder are completely distributed over all  $n$  players. Players in turn ask queries<sup>2</sup>.

Each query consists of a location, a weapon, and a suspect. If player  $i$  asks such a query, then in order, players  $i + 1, i + 2, \dots, n, 1, \dots, i - 1$  must try to disprove the query by showing one of the three cards to player  $i$ . If the player has at least one of the three cards, then they *must* show one of the cards, but can choose which one to show. If they don't have any of the three cards, then obviously they cannot show any card, and must say so. Once one player shows a card to player  $i$ , the query is aborted, and the other players will not be queried. In this way, players gradually learn which other players have which cards, and thereby figure out which three cards are hidden.

You are to write a program that will use the information from queries and responses to figure out as much as possible about the candidates. We will assume here that you are player 1, and can learn only from queries submitted by yourself<sup>3</sup>. Therefore, all the data sets will contain as input only your own queries and the corresponding answers, never the queries by other players.

## Input

The first line contains the number  $K$  of data sets. This is followed by  $K$  data sets, each of the following form:

The first line of each data set contains three numbers  $n, c, q$ , the number of players, cards you received, and queries, with  $2 \leq n \leq 6, 0 \leq c \leq 18$  and  $0 \leq q \leq 1000$ . This is followed by  $c$  lines, each giving the name for one card you hold yourself. Next come the  $q$  queries, each of the following form. The first line of a query is the proposed location, the second line the weapon, and the third the murderer. All of these, as well as the  $c$  cards, will be spelled exactly as given above (including capitalization). This is followed by up to  $n - 1$  lines, the responses by the other players. Each response  $i$  is either the word "Nothing" (if player  $i$  has none of the three cards), or otherwise the name of one of the cards that player  $i$  holds. If it is the name of a card, then the next line begins a new query.

## Output

For each data set, output "Data Set  $x$ :" on a line by itself, where  $x$  is its number. On the subsequent lines, output, one by one, the locations, weapons, and suspects not yet ruled out. Each group should be in alphabetical order. Each data set should be followed by one empty line.

---

<sup>2</sup>To do so, they need to roll dice and move around, but we'll ignore that part here.

<sup>3</sup>While this is not quite true in practice, it simplifies the problem a bit.

## Sample Input/Output

Sample input clue.in

```
1
4 5 3
Ballroom
Dining Room
Candlestick
Gun
Knife
Lounge
Knife
Miss Scarlet
Nothing
Nothing
Lounge
Dining Room
Knife
Professor Plum
Nothing
Nothing
Nothing
Study
Rope
Mrs. White
Rope
```

Corresponding output

```
Data Set 1:
Billiard Room
Conservatory
Hall
Kitchen
Library
Study
Lead Pipe
Wrench
Professor Plum
```

# Problem C: Fingerprints

File Name: fingers.cpp|fingers.java

Input File: fingers.in

## Description

One of the classic ways to find a crime suspect is if they left fingerprints. To be able to use fingerprints to identify a suspect, the police need a large database of known fingerprints, and then compare the one on the crime scene to find the closest match. Here, you are to write a program to do that search for them.

Each fingerprint will be represented by a  $5 \times 5$  black and white bitmap. Black pixels are denoted by 'x', while white pixels are '.'. Thus, fingerprints are represented by 5 strings of 5 characters each, where each character is either an 'x' or a '.'.

The distance between two fingerprints is the number of pixels in which they differ. The best match is the one with the smallest such distance.

## Input

The first line contains two numbers  $n, K$ .  $n \leq 100$  is the number of fingerprints in the police database, while  $K \leq 20$  is the number of crimes you are supposed to solve. This is followed first by the  $n$  fingerprints in the database, each consisting of 5 lines of 5 characters each. Next are the  $K$  crime fingerprints, also each consisting of 5 lines of 5 characters.

## Output

For each of the crime fingerprints, first output "Data Set x:" on a line by itself, where  $x$  is its number. On the next line, output the number (between 1 and  $n$ ) of the best match in the data base. If there are multiple equally good best matches, output all of them in increasing order, on one line, separated by single spaces. Each case should be followed by a blank line.

## Sample Input/Output

Sample input fingers.in

```
3 2
x..x.
x.x..
..x..
x..x.
x.x..
x.xx.
x...x
x..x.
.x.x.
x..x.
x...x
x..xx
.xx..
x.x..
x..x.
x...x
x...x
.x..x
x..x.
x...x
.x..x
x..xx
xx.x.
.x..x
x..x.
```

Corresponding output

```
Data Set 1:
3
Data Set 2:
2 3
```

# Problem D: The Perfect Alibi

**File Name:** alibi.cpp|alibi.java

**Input File:** alibi.in

## Description

One of the more important things for a criminal to have (besides a weapon, a fast car, an army of martial arts freaks, and an entourage) is an alibi: a confirmation that the person was seen in a different place at the time the crime happened. That rules the criminal out as a suspect.

An alibi is usually of the form that some witness says “I saw him/her in place X from time Y to Z.” If the time range includes the time of the crime, and the police have no reason to mistrust the witness, this would prove that the suspect could not have committed the crime. Why would we mistrust a witness? For one, because some other witness claims that the same person was in another place during an overlapping time window. Here, we consider this as the only reason to mistrust a witness. Any witness who is not trusted is completely discarded. To be more precise, if two witnesses  $A$  and  $B$  claim to have seen the same suspect  $X$  in different places during time intervals which overlap, then we treat the input as though  $A$  and  $B$  never existed. (Of course, untrustworthy witnesses don’t prove guilt.)

You are to write a program to narrow down an initial set of suspects by using alibis given by witnesses, but discounting all untrustworthy witnesses.

## Input

The first line contains the number  $K$  of data sets. This is followed by  $K$  data sets, each of the following form:

The first line contains four numbers  $s, w, p, t$ .  $1 \leq s \leq 50$  is the number of suspects,  $1 \leq w \leq 200$  the number of witnesses,  $1 \leq p \leq 50$  the number of possible locations, and  $0 \leq t \leq 1000$  the time of the crime (all are integers).

This is followed by  $w$  lines, each describing one witness statement by giving four numbers  $S_i, P_i, b_i, f_i$ .  $1 \leq S_i \leq s$  is the number of the suspect the witness claims to have seen,  $1 \leq P_i \leq p$  the place at which the witness claims to have been, and  $[b_i, f_i]$  the claimed time interval of the observation. All these numbers are integers.

## Output

For each data set, output “Data Set  $x$ :” on a line by itself, where  $x$  is its number. On the subsequent lines, output, one per line, the suspects who are still under suspicion, i.e., the ones without valid alibis. If all suspects have alibis, output “No suspect.” instead. They should be output in sorted order. Each data set should be followed by an empty line.

## Sample Input/Output

Sample input alibi.in

```
2
3 5 3 12
1 1 0 5
1 2 4 10
1 3 13 19
2 1 0 12
3 3 13 20
1 4 3 7
1 1 0 8
1 1 4 10
1 2 9 13
1 3 11 15
```

Corresponding output

```
Data Set 1:
1
3

Data Set 2:
No suspect.
```

# Problem E: Emergency Response

File Name: response.cpp|response.java

Input File: response.in

## Description

When a crime happens, it is imperative that the emergency police response reach the crime scene as quickly as possible. That allows them to secure as much evidence as possible, perhaps save the victim, and maybe even catch the perpetrator. In order to achieve this, it is often useful to dispatch emergency response vehicles from multiple starting locations, to avoid traffic delays and such. Here, you are to write a program that will calculate when the first of multiple vehicles will reach the crime scene.

The city is described as  $n$  intersections and  $m$  roads. For each road, the starting and ending intersections  $i$  and  $j$  and the travel time  $t(i, j) \geq 0$  are given. When a pair  $(i, j)$  does not appear in the list, that means that there is no direct road from  $i$  to  $j$ . Notice that the time from  $i$  to  $j$  may not be the same as the one from  $j$  to  $i$ . For instance, a street could be a one-way street, or have different traffic patterns in both directions. In addition, you will be given the starting locations of all the cars and the destination, each as an intersection.

## Input

The first line contains three numbers,  $n, m, s$ .  $n \leq 1000$  is the number of intersections,  $m \leq 10000$  the number of roads, and  $s$  the number of subsequent scenarios. This is followed by  $m$  lines, each specifying a road with three numbers: the starting and ending points  $i$  and  $j$  and the travel time  $t(i, j) \geq 0$ , a floating point number.

Subsequently, there are  $s$  scenarios. Each scenario has as its first line two numbers  $c, k$ .  $c$  is the number of the intersection at which the crime happened, and  $k$  is the number of cars dispatched. This is followed by another line with  $k$  numbers on it, the starting intersections of the  $k$  cars, separated by a single space.

## Output

For each scenario, output “Scenario  $x$ .” on a line by itself, where  $x$  is its number. On the next line, output the earliest arrival time of any vehicle, as a floating point number rounded to two decimals. If no vehicle can reach the target intersection, output “Impossible.” instead. Each scenario should be followed by an empty line.

Sample input response.in

```
6 9 2
1 2 3.5
1 3 1.2
3 4 4.9
2 4 0.221
5 4 0.1
5 6 1.3
4 6 1
2 3 0
3 2 5
4 2
1 3
5 1
6
```

Corresponding output

```
Scenario 1:
3.72

Scenario 2:
Impossible.
```

# Problem F: Arsenic and Old Lace

**File Name:** poisoned.cpp|poisoned.java

**Input File:** poisoned.in

## Description

Many murder mysteries — in particular in movies, but even in real life — involve the victim getting poisoned by an evil concoction of chemicals, herbs, radioactive substances, or other things. The clue to discovering the murderer is finding out what rare substances were used. For instance, if the poison were obtained from poison dart frogs, either the murderer or an accomplice would have to have been in South or Central America recently. But this requires figuring out the individual substances mixed together.

In order to do this, the police can draw on an analysis of the poison used, with a list of all the individual substances in it. They also have a list of base products with all their individual substances. The question is then what is the smallest number of base products that could have been combined to explain all the individual substances in the victim.

## Input

The first line is the number  $K$  of input data sets, followed by the  $K$  data sets, each of the following form:

The first line of each data set contains two numbers  $s, b$ , the number of individual substances  $1 \leq s \leq 50$ , and the number of base products  $1 \leq b \leq 20$ . This is followed by  $b$  lines, each describing a base product with a string of  $s$  letters. The  $i^{\text{th}}$  character is 'y' if the base product contains the individual substance  $i$ , and 'n' otherwise. Finally, one more line describes the poison in the victim in the same way.

## Output

For each data set, output "Data Set  $x$ :" on a line by itself, where  $x$  is its number. On the next line, output the minimum number of base substances that together would provide all the individual substances in the poison, but no more. If there is no such combination, output "Impossible." instead. Each data set should be followed by a blank line.

## Sample Input/Output

Sample input poisoned.in

```
2
5 5
ynnnn
nnyyn
ynyyn
nnnny
ynyyy
ynyny
3 3
yyn
nyy
yny
ynn
```

Corresponding output

```
Data Set 1:
2

Data Set 2:
Impossible.
```