acm International Collegiate Programming Contest 2000

South Central USA Regional Programming Contest

Problem and Solutions Index

ACM

ACM Home Page
Intl Prog Contest
South Central US Region

Local Contest

Home
Schedule
Information
Details
FAQ

Results

Report
Final
Attempts
Times
Problems
Movies

Here are the problems and the judges solutions from the contest:

1. Is this Poison? - poison.java
2. Shadows - shadows.c
3. Election - election.cpp
4. Virtual Computer - computer.cpp
5. Sebastion's Problem - baz.java
6. Tournament - tourney.c
7. Sisco's Galaxy - sisco.java
8. Tetris - tetris.c

The currently provided solutions are in their **original** form, reading from a file as opposed to reading from `stdin` and `stdout`. The actual solutions, used to certify the judge output data, were lost on a bad floppy.

Retturn to the Main Page

LSU

Home
Map
Search
ACM Chapter
Computer Science Computing Services

Contest Details

Welcome
Rules
Login
Software
PC^2 instructions

Detailed Schedule

This page last updated 2000/11/16 14:32:41.

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #1: Is this poison?

Introduction

You have recently landed a job with an supercomputing application hosting firm. The firm's primary mission is to provide distributed computing for its customers across its networked supercomputer. The supercomputer consists of many individual computers of varying size (i.e. Mainframes, Servers, and Workstations) and configurations (i.e. OS's, hardware, MIPS). Each of these computers are commonly referred to as processing nodes. Customers submit to a centralized distribution point the programs they want executed and the sets of data files to be processed. Each program is distributed to the processing nodes along with a particular data file. The program is then executed on each node with general security privileges. The types of programs that customers submit are thread intensive, utilize a large amount of inter-process communication, and execute generic OS system calls. The results of the processing are then communicated to the submitter.

The firm's customers are primarily competing bioengineering firms researching the human genome. Each is racing to patent particular genes and gene interactions which they discover. Since the applications are submitted directly by the customers and many customers will be submitting programs for distributed computation at the same time, security of the processing nodes and of individual customer data and source code is of paramount importance.

Your project lead is concerned that some customers may try to interject virus or malicious code into the distribution process, either to crash the processing nodes, steal the source code and data of other customers, or affect the programs of other customers in order to produce erroneous data.

You have been tasked with writing a malicious code scanning program for the distribution subsystem. Its primary functional requirement is to scan programs and data files submitted by customers and compare them to the malicious code library your company maintains. This program will produce reports indicating if malicious code exists in the submitted program and data files.

Here are a the requirements given to you by your lead:

- Malicious code characters always appear in files sequentially without interruption i.e. "the dog" will always appear as "the dog" never as "the cat dog".
- A malicious code character may differ from the signature character, however they will only differ in the character ASCII value not in the amount of characters i.e. "the dog" may be represented as "The dog" or " he dog" or "tHe doc" but not as "the ddog". **SPECIAL CASE: at no time will an EOLN (end of line)/ carriage return character replace an ASCII value of a malicious code character.**
- Malicious code match percentages are determined with the following formula:

Code match percentage = # of characters matching the virus signature

# of characters in the virus signature.

- Code, or data files, are considered CLEAN or POISON.
- "CLEAN" indicates the file does not contain any malicious code matches equaling or exceeding the threshold percentage for any signature. "POISON" indicates that the file does contain malicious code matches exceeding the threshold percentage for at least one signature.
- If more than one instance of malicious code exists, examine them all. If any one exceeds the threshold the code is considered POISON.

Input

Input will consist of up to 100 data sets. Each data set will be correct and follow the following rules:

- A set of malicious code descriptions contains the following:

Elements of a Malicious code description	Description	Example
Malicious code signatures start line	A set of characters of the form MALICIOUS CODE SIGNATURES followed by a new line character.	`MALICIOUS CODE SIGNATURES`
Any number of Code Signatures	Next bullet in the list	

- Code Signature(s) - 0..n instances of the following:

Elements of a Signature	Description	Example
Signature description start line	A set of characters of the form <*x*> (any combination of the following valid characters "a".."z", "A".."Z", "0".."9", & ".") followed by a new line character. <*x*> is the name of the signature.	`RABBIT`
Threshold percentage start line	A set of characters of the form THRESHOLD PERCENTAGE followed by a new line character.	`THRESHOLD PERCENTAGE`
Threshold percentage	A set of characters of the form <*y*> (any combination of the following valid characters "0".."9" & ".") followed by a new line character. <y> is the threshold percentage to 2 decimal places of precision i.e. 100.00, 95.03 etc.	`50.00`
Threshold percentage end line	A set of characters of the form THRESHOLD PERCENTAGE followed by a new line character.	`THRESHOLD PERCENTAGE`
Code Signature start line	A set of characters of the form SIGNATURE followed by a new line character.	`SIGNATURE`
Code signature	A set of characters (1..n - all ASCII characters except EOF and EOLN are valid) followed by a new line character. The entire contents of these lines is the code signature.	`We`

Code Signature end line	A set of characters of the form SIGNATURE followed by a new line character.	SIGNATURE
Signature description end line	A set of characters of the form SIGNATURE followed by a new line character.	RABBIT

Elements of a Malicious code description	Description	Example
Malicious code signatures end line	A set of characters of the form MALICIOUS CODE SIGNATURES followed by a new line character.	MALICIOUS CODE SIGNATURES

- Code or data to scan - 1 instance of the following:

Elements of code file to scan	Description	Example
File start line	A set of characters of the form <z> (any combination of the following valid characters "a".."z", "A".."Z", "0".."9", & ".") followed by a new line character. <z> is the name of the file.	XCHROMOSOMECALC1

Data start line	A set of characters of the form DATA followed by a new line character.	DATA

Data	A set of characters (1..n - all ASCII characters except EOF are valid) followed by a new line character. The entire contents of these lines is the data or program file to be scanned.	Class calc {
		public calc() {
		Wefg6me3to 632edcM
		regsdfal p3455agdisg fogiest.
		System.out.println("cmo");
		While heM; egimm;
		ing; System
		.out.println("con\ntest");
		}
		}

Data end line	A set of characters of the form DATA followed by a new line character.	DATA
File end line	A set of characters of the form <z> (any combination of the following valid characters "a".."z", "A".."Z", "0".."9", & ".") followed by a new line character. <z> is the name of the file.	XCHROMOSOMECALC1

- Blank line - separating the start of another data set from the previous data set.

Output

For each data set, output will consist of a set of characters of the form "<z> <r>" where <z> is the name of the data file which was scanned (any combination of the following valid characters "a".."z", "A".."Z", "0".."9", & ".") and <r> is a set of characters of the form "CLEAN" or "POISON" followed by a new line character. "CLEAN" indicates the file does not contain any malicious code matches equaling or exceeding the threshold percentage for any signature. "POISON" indicates that the file does contain malicious code matches exceeding the threshold percentage for at least one signature.

Sample Input

```
MALICIOUS CODE SIGNATURES
MALICIOUS CODE SIGNATURES
XCHROMOSOMECALC1
DATA

Class calc {
public calc() {
Wefg6me3to 632edcM
regsdfal p3455agdisg fogiest.
System.out.println("cmo");
While heM; egimm;
ing; System .out.println("con\ntest");
}
}
DATA
XCHROMOSOMECALC1

MALICIOUS CODE SIGNATURES
RABBIT
THRESHOLD PERCENTAGE
50.00
THRESHOLD PERCENTAGE
SIGNATURE
```

```
We
SIGNATURE
RABBIT
MALICIOUS CODE SIGNATURES
XCHROMOSOMECALC1
DATA

Class calc {
public calc() {
Wefg6me3to 632edcM
regsdfal p3455agdisg fogiest.
System.out.println("cmo");
While heM; egimm;
ing; System .out.println("con\ntest");
}
}
DATA
XCHROMOSOMECALC1

MALICIOUS CODE SIGNATURES
RABBIT
THRESHOLD PERCENTAGE
70.00
THRESHOLD PERCENTAGE
SIGNATURE
Welcome to the ACM regional programming contest.
SIGNATURE
RABBIT
FROG
THRESHOLD PERCENTAGE
85.05
THRESHOLD PERCENTAGE
SIGNATURE
Thats the way uh-hu uh-hu I like it..
SIGNATURE
FROG
MALICIOUS CODE SIGNATURES
PROTEINFOLDING.2
DATA

Public Class {
format c:
```

```
Thats the ACM regional hu I like it..
}
DATA
PROTEINFOLDING.2
```

Sample Output

```
XCHROMOSOMECALC1 CLEAN
XCHROMOSOMECALC1 POISON
PROTEINFOLDING.2 CLEAN
```

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #2: Shadows

Introduction

A LSU physics researcher who we will call "Sparky" was very successful in procuring funding for the physics department because of his breakthrough research into new 3-dimensional crystalline storage technology. Though a brilliant physicist, Sparky's organizational habits were lacking. His office was strewn with stacks of crystal pictures, and most were not labeled in any way other than to note the orientation of the crystal.

Unfortunately for the physics department, Sparky asphyxiated from laughter after a night grading midterms for his Physics For Non-Majors class. You are the lucky new non-tenured professor that has been hired by the Physics departement to decipher Sparky's research and keep that grant money flowing.

In Sparky's now vacant office, there is a mess of picutures all about. You know that he was taking three pictures of each crystal he grew (one picture down each primary axis: X,Y,Z). You also know that the crystals grew in a liquid suspension and therefore don't necessarily consist of a single continuous mass.

As a first step in organizing the mess you decide to take the pictures three at a time (one down each axis) and see whether or not they could be pictures of the same crystal. Of course, you know Sparky always took pictures of cubical areas, so you pick out pictures of the same dimensions before examining them.

Here are some other useful things that you know about the pictures:

- All the pictures are black-and-white.
- Every picture is of resolution NxN (in other words, each picture is a square).
- Each picture was created by placing the crystal on top of the film and shining a light through the crystal. The result was that areas of the film hit directly by the light (without passing through the crystal) are bright. Any light striking the crystal is reflected away from the film. The resulting picture is therefore a silohuette of the crystal projected down a particular axis.
- The crystal grows in a very regular way. It creates small cubical areas precisely 1x1x1 that may be adjacent to one another or that may grow separately. In

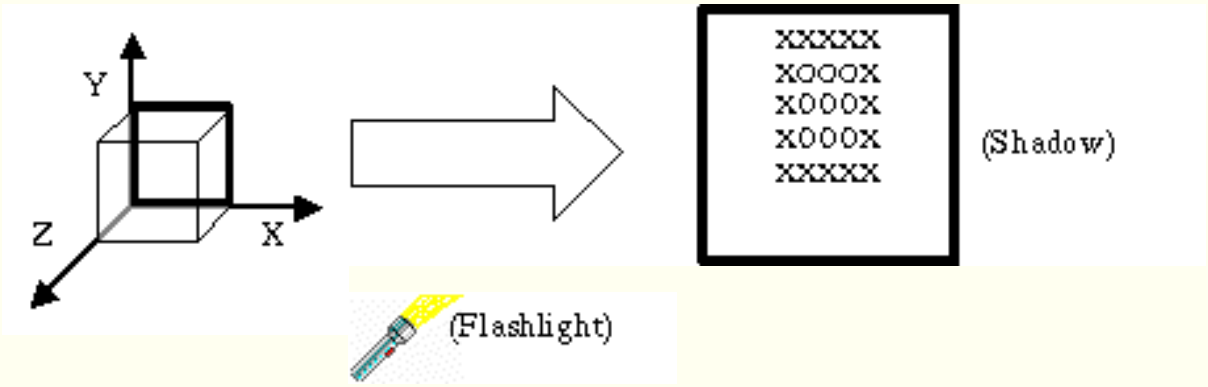
essence, the crystal is a NxNxN matrix where at any coordinate in the matrix a piece of crystal may or may not reside.

Input

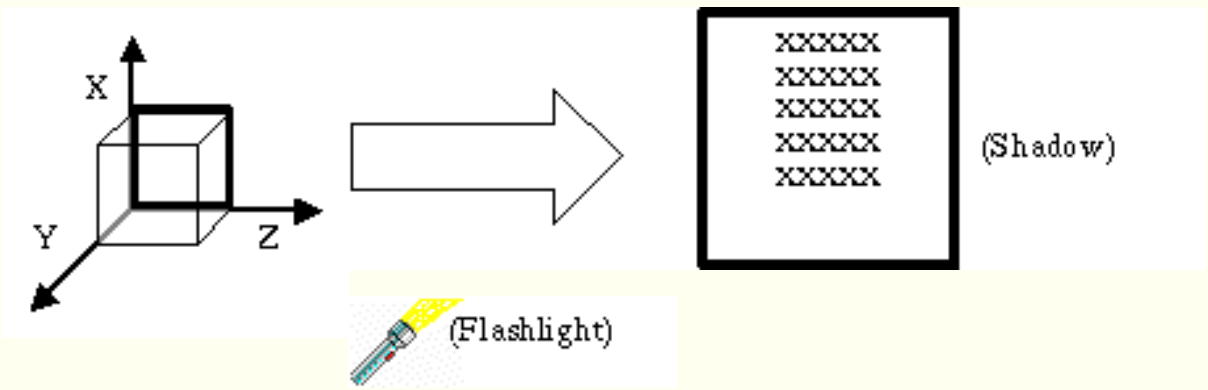
Input to this problem will consist of a (non-empty) series of up to 100 data sets. Each data set will be formatted according to the following description, and there will be **no blank lines** separating data sets.

A single data set has five components:

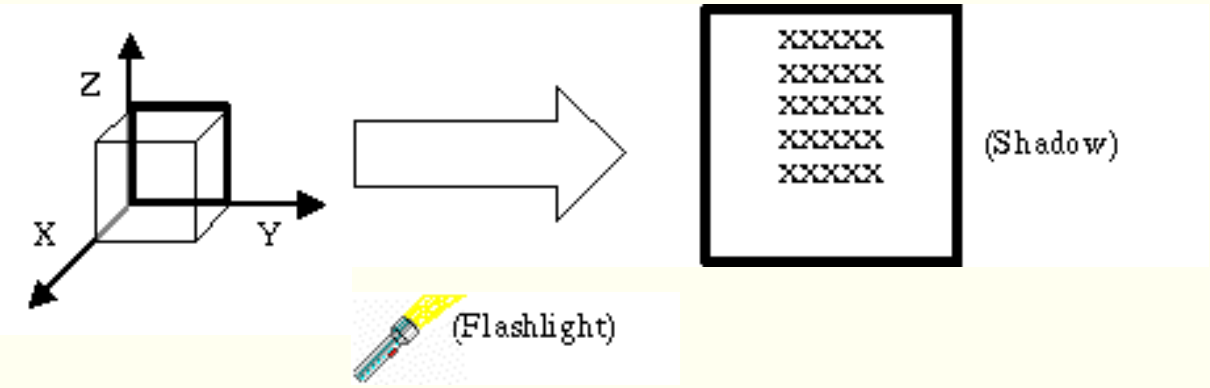
1. *Start line* - A single line, "`START <cube_size>`" where <cube_size> is a positive integer in the range 1-10, **inclusive**, that gives the height/width/depth for the cube (fortunately, height/width/depth are all the same for cubes, so you only need one number). For example:
   `START 5`
2. *Shadow 1* - A shadow down the Z axis. Shadows consist of X's and O's (letter not number), where an X represents a one unit square area of shadow and an O represents a similarly sized illuminated area. There is no white-space leading or following any of the lines. For example:



3. *Shadow 2* - A shadow down the Y axis



4. *Shadow 3* - A shadow down the X axis



5. *End line* - A single line, "END"

Output

For each data set, there will be exactly one line of output. This line will simply be the word "YES" or the word "NO" (all caps with no whitespace leading or following).

A "YES" line will appear only if there exists *some* solid 3-dimensional object that could project the given silhouettes. (In other words, these could all be pictures of the same crystal).

A "NO" line will be output for all data sets that fail to meet the criteria for a "YES" line.

Sample Input

```
START 1
O
O
O
END
START 3
XXX
XOX
XOX
XXX
XXX
XXX
XXX
```

```
XXX
XXX
END
START 7
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
XXXXXXX
OOOOOOO
OOOOOOO
OOOOOOO
OOOOOOO
OOOOOOO
OOOOOOO
OOOOOOO
XOOOOOO
OXOOOOO
OOXOOOO
OOOXOOO
OOOOXOO
OOOOOXO
OOOOOOX
END
```

Sample Output

```
NO
YES
NO
```

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #3: Election

Introduction

After numerous recounts of the presidential votes in Florida, candidates George W. Bush (Republican) and Al Gore (Democrat) are unconvinced of the accuracy of the results. Each candidate is convinced that he should receive Florida's electoral votes needed to win the presidency and no amount of recounting the ballots will determine a clear winner. Also, it was revealed that some of the volunteers forgot to turn in several ballot boxes, and are currently unavailable on vacation at Disney World.

The U.S. population is justly demanding a resolution to the problem to avoid a third term under Clinton. Political analysts insist that they can determine which candidate the state will be awarded to, based on the income levels of its districts' residents. Jeb Bush, the governor of Florida, has asked you to write a program that will determine which candidate will win Florida, and ultimately the election. The future of the country rests in your hands.

The program will be based on the electoral college system, extended to the district level. Each district is worth a certain number of electoral votes, with the winner of the district determined by the median income of its residents. Exit polls have revealed that if the median income of the district is greater than a certain level, it will be awarded to the Republican candidate. A simple majority of electoral votes is needed to win the entire state (a tie will result in the deciding vote being cast by Jeb, resulting in a Republican victory).

Input

Input will consist of a non-zero number of independent data sets containing election information. Each election data set will conform to the following:

1. The first line of a data set will be 'START'.
2. The next two lines (this is a run-off election, no third-party candidates) will contain the names of the Democratic candidate coming first.

3. The fourth line will contain the number of districts.
4. The fifth line will contain the target median income level. A district with a median income greater than the target will have its electoral votes go to the Republican candidate. All other electoral votes will go to the Democratic candidate.
5. Each of the next lines will contain a district's name, its median income level (formatted as a simple integer), and its number of electoral votes (also an integer), each separated by a comma.
6. The last line of a data set will be 'END'.

There will be no blank lines in the input, but there will be an endline just before the end of the input.

Output

The output will be the results of the elections from the input. For a given election set, the output will consist of a line containg the winning candidates' name and his number of electoral votes, separated by a comma (there will be no spaces on either side of the comma):

```
George W. Bush,96
```

There will be no blank lines separating output sets.

Sample Input

```
START
Al Gore
George W. Bush
12
39000
North,50000,10
Northnortheast,30000,18
Northeast,40000,11
East,40000,15
Southeast,43000,13
Southsoutheast,52000,14
South,47000,9
Southsouthwest,25000,20
Southwest,25000,24
```

```
West,60000,12
Northwest,39000,11
Northnorthwest,35000,12
END
START
Bleed N. Hart
Pub Lee Kahn
4
37000
North,25000,40
East,35000,20
South,40000,30
West,45000,20
END
```

Sample Output

```
Al Gore,85
Bleed N. Hart,60
```

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #4: Virtual Computer

Introduction

James recently purchased a new palm-top computer at a discount store. To get a really good price, he decided to have the free operating system, "JameSIX", installed instead of the more popular and expensive "Windwoes" operating system. One drawback to the free OS is that there are many bugs, but James is a savvy computer programmer and thinks it worth the risk.

A few days later, James is attempting to play "Sweeper" when all of a sudden the screen flashes and some strange output appears. Angered by this interruption to his favorite game, James decides to use his vast programming knowledge and attempt to debug the program in the free debugger, JDB.

James has found a suspicious piece of code. He knows the values of all variables, and he is attempting to determine if the code fragment was the one that produced the output that interrupted his high-score attempt.

The "JameSIX" Assembly language in which the program was written has a very simple set of commands:

Command	Explanation
`START <progname>`	The first statement of a program, where `<progname>` is the program's integer identifier Example: `START 21`
`SET <x> <value>`	Will set variable `<x>`'s value to `<value>` Example: `SET a 9` is equivalent to `a = 9` Note: `<x> <y>` can only be integers.
`STORE <x> <y>`	Will set variable`<x>`'s value to the value of variable`<y>` Example: `STORE b a` is equivalent to `b = a`
`MULT <x> <y>`	Equivalent to variable `<x>` = variable `<x>` * variable `<y>` Example: `MULT b a` is equivalent to `b = b * a`

`SUB <x> <y>`	Equivalent to variable `<x>` = variable `<x>` - variable `<y>` Example: `SUB b c` is equivalent to `b = b - c`
`DIV <x> <y>`	Equivalent to variable `<x>` = floor(variable `<x>` / variable `<y>`) except in the case where variable `<y>`'s value is `0`. If variable `<y>`'s value is equla to `0`, then the statement has no effect on the value of variable `<x>` or variable `<y>` or any other variable within the program (i.e. `<x> = <x>`). Example: `DIV b c` is equivalent to `b = b - c`
`ADD <x> <y>`	Equivalent to variable `<x>` = variable `<x>` + variable `<y>` Example: `ADD b c` is equivalent to `b = b + c`
`PRINT <x>`	Prints the value of the variable `<x>` followed by a carriage-return Example: `PRINT b`
`GOTO <n>`	Execution moves to the nth line of the program. The first line is line number 1. Example: `GOTO 10`
`END`	This statement ends the current program. Example: `END`

And one conditional command (non-simple):

Command	Explanation

`IF <x> IS <value> THEN <simple_command>`	Where `<simple_command>` is any of the previously mentioned simple commands. `<simple_command>` is only executed if the value of the variable `<x>` is equivalent to the integer `<value>` (this is not a variable).
	Example: `IF b IS 3 THEN ADD b c` is equivalent to
	`if b = 3 then`
	`  b = b + c`
	`endif`

NOTE: All variables are of type integer.

Input

Input will consist of a non-zero number of 'programs', each of which are completely independent of one another. Each program will conform to the following:

1. The first statement (line 1 of each program) will be '`START <progname>`' (See above for clarification on `START` command).
2. The last statement will be '`END`'.
3. All variables will be initialized with a '`SET`' statement before being used in any other statement.
4. Program ID numbers will be unique.
5. Line numbering resets each time a new program begins (each '`START`' statement).
6. All variables are single lowercase letters (`a`, `b`, `c`, `...`, `z`).
7. All variable values will be within the following range throughout the program; `-1000 <= a <= 1000` where a is the variable i.e. `ADD x y` will never result in a value outside of the range.

Arguments to individual statements will always be separated by exactly one space, and there will not be any blank lines in the input file.

Output

The output will be the results of the programs in the input file. For a given program, you will print to the output file:

1. A single line containing '`START <progname>`' exactly like the first line of the program when execution of a new program begins
2. A single line with the value currently stored in the variable `<x>` when a '`PRINT <x>`' statement is executed.

There will be no blank lines separating output sets.

Sample Input

```
START 1
SET b 10
PRINT b
END
START 2
```

```
SET b 10
SET a 1
SET c 0
STORE c a
MULT c b
PRINT c
END
START 10
SET a 1
SET b 2
SET c 3
SET d 4
SET e 5
STORE e a
ADD a a
PRINT a
MULT c b
IF e IS 8 GOTO 13
GOTO 7
PRINT c
END
```

Sample Output

```
START 1
10
START 2
10
START 10
2
4
8
16
48
```

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #5: Sebastian's Problem

Introduction

Once upon a time there lived a programmer with the potential to be the best Java programmer the world had ever seen, and his name was Sebastian. Unfortunately for Sebastian, he was born several decades prematurely and was forced to store all of his programs on punchcards. If you do not know what a punchcard is ask one of your professors after the contest. We are sure she will be able to tell you many horror stories about punchcards and shoe boxes.

One day Sebastian had to turn in his senior project, a program that calculated pi to a billion decimal places. On his way to deliver his program to the grader, he was suddenly overtaken by a vicious case of scurvy (pizza just doesn't have that much vitamin C). In his mad dash to the nearest lavatory, he dropped the box of punchcards containing his program, and the cards scattered everywhere.

After he had picked up all the cards, and counted them, he realized that one was missing. Luckily Sebastian had taken the time to record his program card-by-card in a notebook at the dorm. This will allow him to replicate the missing card and still turn his project in on time. The problem was that the cards were all jumbled and he didn't know which card to replicate.

Here are a few things that Sebastian knows about his cards:

- Sebastian's program had at least 1 card before he dropped it.
- There are *n* cards numbered from 1 to *n*.
- Exactly one card is missing.
- No cards have duplicate numbers.

Please write a program to help Sebastian determine what card number his needs to replicate so that he can turn-in his project on time.

Input

Input will consist of up to 100 data sets. Each data set will consist of:

1. *Start line* - A single line of the form "START *n*" where *n* is the number of punch cards in the program. No program will consist of more than 10,000 cards.
2. *Card list* - A list of *n*-1 card numbers in the range 1 to *n* (inclusive). Each card number will appear on its own line with no leading or following whitespace.
3. *End line* - A single line of the form "END".

Output

For each data set, output will consist of a single number on its own line representing the card that was missing from the *Card list* of that data set.

Sample Input

```
START 1
END
START 5
1
2
4
5
END
START 20
13
14
17
1
2
3
4
15
16
5
18
20
6
10
```

```
11
12
7
8
9
END
```

(Note: these samples are **VERY** small compared to some of the data sets your problem will be judged against and are abbreviated due to the number of trees that would have to be destroyed to provide a large example. The limit is 10,000 cards per data set.)

Sample Output

```
1
3
19
```

2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #6: The Tournament

Introduction

My Uncle Lester is known throughout Lafayette Parish as the luckiest man around. He has picked the winners of the last 7 NCAA Basketball Championships. During Mardi Gras last year, I got Uncle Lester to explain exactly how he is able to consistently pick the winners of this event. He is convinced that if you rate teams' offense and defense on a scale of 1 to 10 and then multiply their offensive rating by 5 and subtract double their opponents defensive rating, you will have calculated that teams score if that team played their opponent. (Keep in mind that if a score is not an integer, then that score will be rounded to the nearest whole number.) I told Uncle Lester that this could result in negative scores and he told me that was true, but the eventual outcome is accurate. Whoever has the greatest score will win the game and continue on to the next round.

The result of team A playing team B is calculated as follows:

Team	Offensive	Defensive
A	w	x
B	y	z

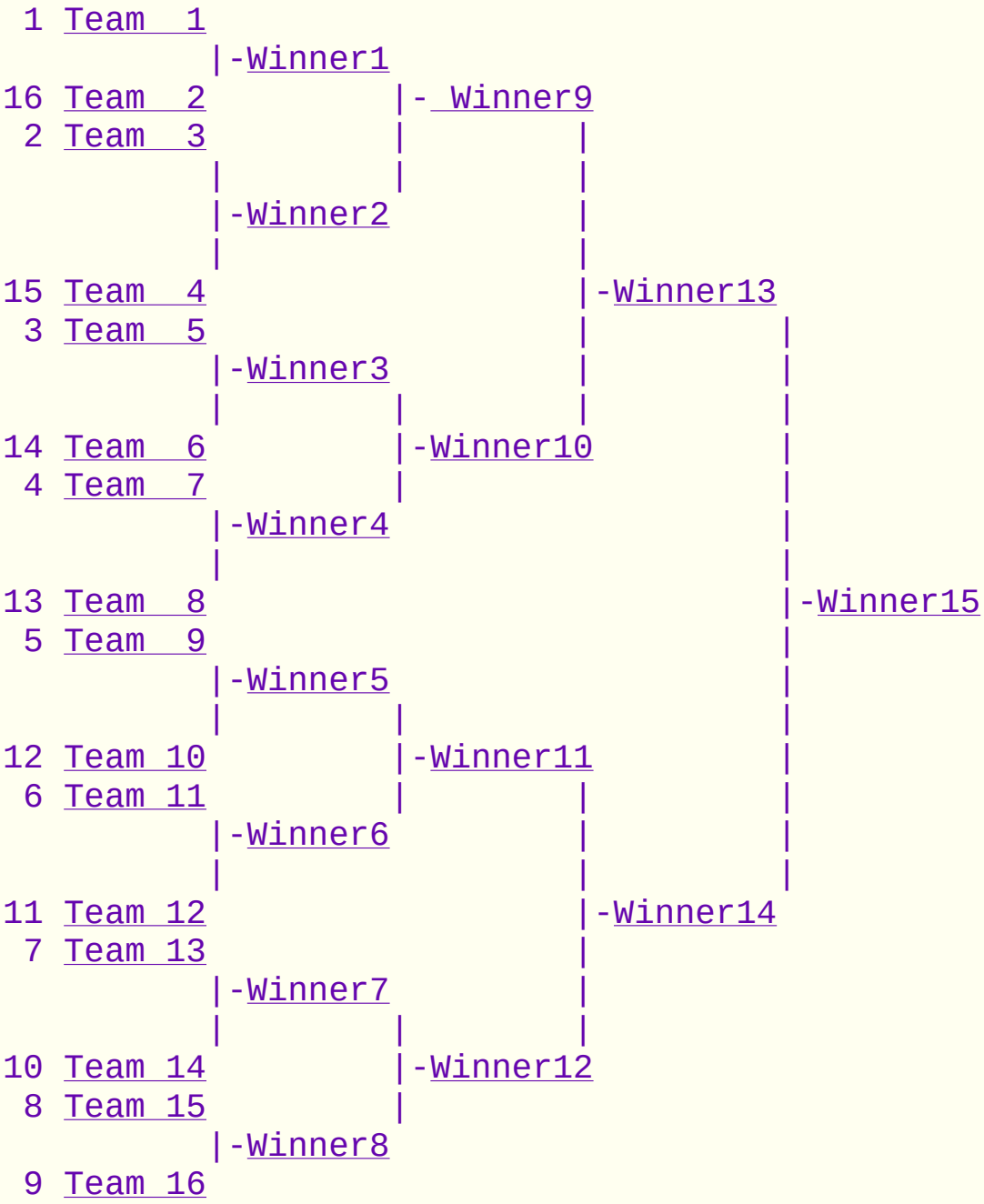
Score of team A = 5*w - 2*z

Score of team B = 5*y - 2*x

If one team beats the other team by more than 10 points, the winning team's offensive rating will increase by .5. A tie score will result in the winner being the team with the mascot name which would appear first in alphabetical order between the two teams.

As you may know, the NCAA seeds all teams in the tournament. They seed the teams according to their offensive rating. The higher the offensive rating is, the lower the seed is. If more than 1 team has the same offensive rating, then the school mascot names are sorted in alphabetical order with the first team in the list getting the next lowest seed number available. For example, if in determining seed 3, a team with

mascot name Bananas has the same offensive rating as a team with mascot name Mangoes then Bananas would be awarded seed 3 position. The tournament bracket is laid out where the highest seeded team plays the lowest seeded team, the second highest seeded team plays the second lowest seeded team, and so on. The winner of the first two games play each other and so on.



With all of Uncle Lester's theories, and the NCAA's rules for tournament seeding, who will win the next NCAA Basketball Championship, and what will the final score be?

Input

Input will consist of n data sets in the following format:

- A line of the form <x> followed by a new line. <x> is the number of schools in the tournament.
- <x> lines of the form <school name>,<school mascot>,<offensive rating>,<defensive rating> followed by a new line.
- There will always be enough schools to fill-out a complete bracket (i.e. 2^n teams, with n between 1 and 4, inclusively). School name and school mascot will consist solely of alphabetic characters (no spaces).

Output

The output will be the name of the tournament champion and the score of the final game followed by a new line. The final score will be the two scores separated by a minus sign. The champion's score will be first in the sequence.

Sample Input

```
16
LOU,Snorks,7,8
LSU,Tigers,9,6
OU,Sooners,6,7
TAM,Aggies,4,9
UT,Longhorns,8,3
BU,Cubs,4,4
ACM,Hackers,8,4
IEEE,Techs,7,7
MIA,Soldiers,3,3
MIB,Aliens,6,9
TUG,Warriors,3,8
TNG,Trekkies,4,4
CIA,Agents,5,7
FBI,Gmen,6,6
WWF,Wrestlers,3,5
MAY,Flowers,2,5
4
IBM,Judges,5,5
UPS,Drivers,7,3
```

```
TKO,Fighters,8,4
YMCA,Villagers,7,4
```

Sample Output

```
LSU 34-26
TKO 35-27
```

ACM South Central Region 2000 Programming Contest

Louisiana State University

Problem #7: Sisco's Galaxy

Introduction

The Dominion War has ended and Benjamin Sisco has convinced the Wormhole Aliens to allow the Federation unlimited access to the wormhole that connects Alpha Quadrent and Delta Quadrent. In his discussions with the aliens, Sisco learned that there are other Wormholes that exist throughout the galaxy. He has also convinced the aliens to identify the locations of these wormholes and allow the Federation access to them.

The galaxy has now been fully mapped, and The Federation has determined that a new navigational system needs to be built to take advantage of the new charts. Of particular interest to The Federation is how long a particular trip will take. Your team has been tasked with writing that part of navigational system.

The map of the galaxy is 2-dimensional and has been split up into a regular grid of sectors. The first sector is sector 1. Sectors are numbered left to right, bottom to top. It takes one day to travel from one sector to another. It also takes one day to travel through a wormhole from one sector to another. Travel can take place by three types of moves, each of which uses one day:

1. A horizontal move (left or right)
2. A vertical move (up or down)
3. A move that traverses a worm hole from one of the endpoints to the other endpoint.

You can **not** travel to sectors diagonally.

Your specific task will be to construct a system that can determine the minimum number of days it will take to get from a specified starting sector to a specified target sector.

Input

The input will consist of a non-empty series of input sets. Each input set will follow the following format:

```
<Size of the galaxy>

<WormholeName>, <Wormhole endpoint #1>, <Wormhole endpoint #2>
.
.
.
Trip, <Sector the trip begins in>, <Sector the trip ends in>
```

The size of the galaxy is in the format **<Number1>x<Number2>**. **Number1** and **Number2** may be equal, but they do not have to be equal. **Number1** represents the number of *rows*, **Number2** represents the number of *columns*. Both **Number1** and **Number2** will be greater than 0 and less than or equal to 100.

The list of Wormholes is of variable length (0-20 wormholes). Wormholes can be traversed in either direction. **<WormholeName>** may contain alphanumeric characters as well as spaces and underscores.

The single Trip line of the input set specifies the start and end sectors for the trip that the system should chart a course for.

All commas in the input are followed by exactly one space.

Each input set will be separated from the next with a single blank line. There will not be a blank line at the beginning of the input, but there will be an endline just before the end of input.

(Note: The below corresponds to the first example in the sample data. Wormholes exist such that you can travel from 3 to 5 and from 42 to 51.)

91	92	93	94	95	96	97	98	99	100
81	82	83	84	85	86	87	88	89	90
71	72	73	74	75	76	77	78	79	80
61	62	63	64	65	66	67	68	69	70
51	52	53	54	55	56	57	58	59	60
41	42	43	44	45	46	47	48	49	50
31	32	33	34	35	36	37	38	39	40
21	22	23	24	25	26	27	28	29	30
11	12	13	14	15	16	17	18	19	20
1	2	3	4	5	6	7	8	9	10

Output

For each input set, there will be a single line in the output. This line will indicate the minimum number of days (>= 0) it will take to get from the starting sector to the ending sector and will have the following format:

Trip from <start sector> to <end sector> takes <days> days

Sample Input

```
10x10
Wormhole1, 3, 5
Wormhole2, 21, 50
Wormhole3, 90, 100
Trip, 1, 49
100x100
```

```
The OSU Wormhole, 1, 5
The OU Wormhole,  39, 2005
The LSU Wormhole, 3021, 9765
The UL-LA Wormhole, 50, 2005
The Baylor Wormhole,  2205, 3001
Trip, 45, 3021
100x10
The quick trip, 56, 67
Trip, 55, 67
```

Sample Output

```
Trip from 1 to 49 takes 4 days
Trip from 45 to 3021 takes  29 days
Trip from 55 to 67 takes 2 days
```

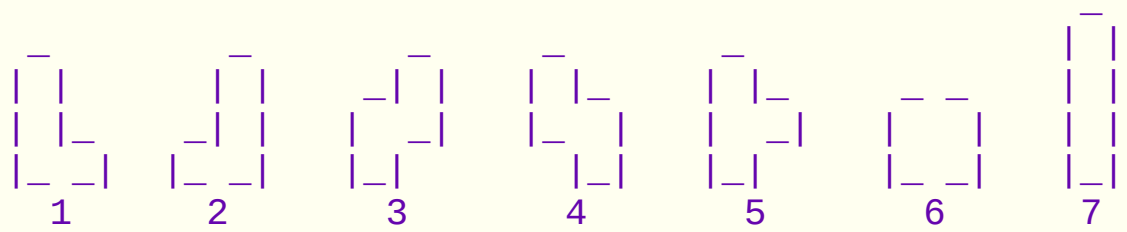
2000 ACM South Central Regional Programming Contest

Louisiana State University

Problem #8: Tetris

Introduction

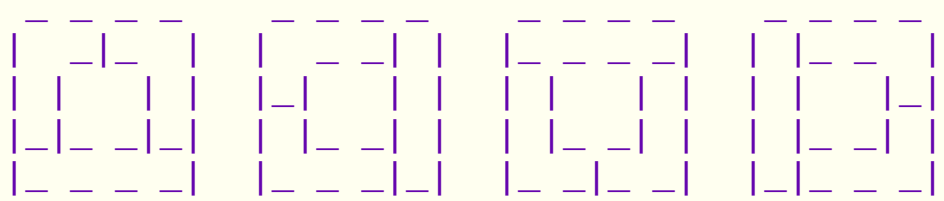
Uncle Jacques has purchased some colorful floor panels for the dance floor of his club. Every Friday his employees will rearrange the panels so that the disco dance floor will have a new and exciting look. He has purchased a large supply of panels with seven different shapes. Each panel consists of four 1 foot squares as depicted below:



Uncle Jacques needs to know if his dance floor can be completely covered using these panels. He also needs to know how many different patterns his employees can create on the dance floor. The panels can be rotated in 90-degree increments, they cannot overlap, they cannot extend past the edge of the dance floor, and they must cover the dance floor completely.

Your first job is to determine whether the dance floor can be completely covered. Your second job is to determine the number of different patterns that can be created by rearranging the panels. Patterns are considered distinct if the borders between panels are different. For example, for a 4 foot by 2 foot dance floor, there are 4 distinct patterns:

Rotations of patterns are also considered distinct. For example, the following four patterns are distinct:



Input

Your program must accept the length and width of the dance floor in feet. There can be any number of dance floor measurements in the input. The dance floor will not be larger than 36 square feet.

Output

There will only be one line of output for each set or measurements. Your program must return the number of distinct patterns, which completely cover the dance floor. If the floor cannot be completely covered, your program must return zero.

Sample Input

```
2 4
3 3
3 4
4 4
```

Sample Output

```
4
0
23
117
```