Carnegie Mellon University
Spring Programming Competition

March 29, 2008

You can program in C, C++, or Java; note that the judges will re-compile your programs before testing.

Your programs should read the test data from the standard input and write results to the standard output; you should not use files for input or output.

All communications with the judges should be through the PC^2 environment.

Each problem's estimated difficulty for a typical and representative participant is shown, using the following scheme, next to each problem's name on the title line. This guidance is only an estimate based on many, possibly incorrect, assumptions. Take it as only one, of many, factors contributing to your estimate of your own team's ability to solve the problem. Remember: Your mileage may vary!

(☺ ☺ ☺ ☺ ☺)	Very Challenging
(☺ ☺ ☺ ☺)	Challenging
(☺ ☺ ☺)	Nuances or subtlety, or upper level math/algo/problem solving
(☺ ☺)	Some nuances or library or algorithmic knowledge required
(☺)	Straight-forward, few if any nuances

For example, consider hypothetical "Problem Y" and "Problem Z", as shown below. "Problem Z" would be estimated a "Very Challenging" problem, whereas "Problem Y", would be estimated to be "Straight-forward".

Problem Y: Fly High! (☺)

Problem Z: Smack 'em down (☺ ☺ ☺ ☺ ☺)

Problem A: Welcome (☺)

Welcome to Hogwarts School of Witchcraft and Wizardry, where Harry Potter and his friends study the art of magic. While solving the competition problems, you will learn new facts about Hogwarts, which have not been revealed in the books and movies about Harry Potter.

The first problem allows you to earn a credit for implementing a Hogwarts version of "Hello World"; specifically, you should write a program that welcomes professors, students, and visitors who enter the school.

Input

The input includes multiple test cases; each case is on a separate line, and the number of cases is at most 20. A test case is a string that represents the name of a professor, student, or visitor, which may include upper-case letters, lower-case letters, hyphens, and spaces; its length is between 1 and 80 characters, including spaces. The last input line is a single period ("`.`"), which does not represent a test case.

Output

For each test case, the output is a single line. If an input line begins with the word "`Professor`", the respective output is "`Welcome, Professor`". If it begins with "`Ms`" or "`Mr`", the output is "`Welcome, Visitor`". Finally, the output in all other cases is "`Welcome, Student`". Your program should process its input in a case-insensitive way; for example, if an input line begins with "`professor`" or "`pRoFeSsOr`", the output should be "`Welcome, Professor`".

Sample input

```
Professor Albus Percival Wulfric Brian Dumbledore
pRoFeSsOr Minerva McGonagall
ProfessorSnape
Professor
Professo
Prof Flitwick
Ms Molly Weasley
mR
Harry Potter
.
```

Sample output

```
Welcome, Professor
Welcome, Professor
Welcome, Professor
Welcome, Professor
Welcome, Student
Welcome, Student
Welcome, Visitor
Welcome, Visitor
Welcome, Student
```

Problem B: Magic strings (☺)

When young witches and wizards begin their studies at Hogwarts, they first learn how to recognize words and sentences that represent spells. Fortunately, the rule for recognizing spells is quite simple; specifically, a student has to check whether a given text contains any of the basic magic words, such as "hocus," "pocus," and "abracadabra." If any of such words is a substring of a given string, then this string is a spell. On the other hand, if a string does not contain any such words, then it has no magic power.

Your task is to write a program that helps novice magicians to recognize spells; specifically, it should input a dictionary of basic magic words and a list of strings, and then determine which of these strings contain magic words.

Input

The input includes two lists of strings, where each list contains between 1 and 100 distinct strings. The first list includes all basic magic words, one per line, where each word is a string of lower-case letters; the length of each word is between 1 and 20. The last line of the list is a single period ("`.`"), which does not represent a word. The second list includes strings of lower-case letters, one per line, which may or may not be spells; the length of each string is between 1 and 80 characters. The last line in the second list is also a single period ("`.`"), which does not represent a string.

Output

For each string in the second list, the output is a single line. If the string includes any magic word, the output is "`magic`"; otherwise, it is "`non-magic`".

Sample input

```
hocus
pocus
abracadabra
verylongmagicword
.
hocus
hocuspocus
abracabracadabradabra
hocupocu
stringcontainingverylongmagicwordinthemiddle
.
```

Sample output

```
magic
magic
magic
non-magic
magic
```

Problem C: Hogwarts houses (☺☺)

The students at Hogwarts are divided into four houses, which bear the names of their founders: Gryffindor, Ravenclaw, Hufflepuff, and Slytherin. The houses compete throughout the school year, by earning and losing points for various events, and the house with most points wins the House Cup at the end of the year. Your task is to write a program that computes the total points earned by each house, and determines the relative standings of the houses.

Input

The input includes only one test case, which consists of four lists of earned points, corresponding to the four houses. The first line in a list is a house name, which is `Gryffindor`, `Ravenclaw`, `Hufflepuff`, or `Slytherin`. The other lines are integer values representing earned and lost points, one value per line; each value is either between `1` and `100` (for earned points), or between `−1` and `−100` (for lost points). The number of integers in a list is between 1 and 1000, and the last line of the list is 0. The last line of the input, after the last list, is a single period ("`.`").

Output

The output consists of four lines; each line includes the name of a house, followed by a single space, and the total points earned by this house. Note that, if the total lost points exceed the total earned points, then the total number of points is negative. The output should show the houses in the order of their relative standings; that is, the first line is the winner of the House Cup, the second line contains the house that came second, and so on. If several houses have earned the same number of points, your program should break this tie by sorting them alphabetically.

Sample input

```
Gryffindor
−1
10
−5
20
0
Ravenclaw
40
−50
5
0
Hufflepuff
−1
−2
−2
0
Slytherin
20
0
.
```

Sample output

```
Gryffindor 24
Slytherin 20
Hufflepuff −5
Ravenclaw −5
```

Problem D: Arithmancy (☺ ☺ ☺ ☺)

The science of Arithmancy is the study of magic based on natural numbers. The concept of *magic natural numbers* is fundamental in Arithmancy, and its role is similar to that of prime numbers in arithmetic; it is based on the following recursive definition.

- The number 0 is not magic, whereas the number 1 is magic.
- For $n > 1$, the number n is magic if we cannot represent it as the sum of either two or three distinct magic numbers.

For example:
- 2 is magic
- 3 is not magic because 3 = 1 + 2
- 4 is magic
- 5 is not magic because 5 = 1 + 4
- 6 is not magic because 6 = 2 + 4
- 7 is not magic because 7 = 1 + 2 + 4
- 8 is magic

Since many numeric spells are based on this concept, magicians often need to determine if a given number is magic; your task is to write a program that identifies such numbers.

Input

The input includes multiple test cases, where each case is an integer between 1 and 1000, on a separate line. The number of test cases is between 1 and 20; the last input line is `-1`, which does not represent a test case.

Output

For each test case, the output is either "`magic`" or "`non-magic`", on a separate line.

Sample input

```
2
3
4
5
-1
```

Sample output

```
magic
non-magic
magic
non-magic
```

Problem E: Astrology (☺ ☺ ☺ ☺)

The students at Hogwarts study Astrology as part of the Divination class, where they learn to evaluate the probabilities of future events. The purpose of Astrology is to identify patterns in the locations of stars, and use these patterns to forecast future developments. When a student analyzes a specific star pattern, she first has to find the area of the largest triangle formed by any three stars, and then compare it with the sizes of other pattern elements. Your task is to write a program that finds the largest triangle in a given pattern.

Input

The input includes multiple test cases, which correspond to different star patterns; the number of cases is at most 20. A test case is a list of stars, each on a separate line; the description of a star consists of two integers between 1 and 1000, separate by a single space, which represent its coordinates. The number of stars in a pattern is between 3 and 100, and they all have distinct coordinates. The last line of each test case is "−1 −1", which does not represent a star. The last line of the input, after the last test case, is also "−1 −1".

Output

For each test case, the output is a real value with exactly one digit after the decimal point, which represents the area of the largest-area triangle in the pattern. If all stars in the pattern are on the same straight line, then the output should be "0.0".

Sample input

```
1 1
10 10
1 10
10 1
−1 −1
1 1
10 10
100 100
1000 1000
−1 −1
−1 −1
```

Sample output

```
40.5
0.0
```

Problem F: Defense against dark arts (☺ ☺ ☺ ☺ ☺)

The Hogwarts program includes several classes in the defense against dark arts, where young magicians learn to defend themselves from evil wizards and magical monsters. In particular, the students have to learn how to fight dementors, which are flying monsters that sometimes attack wizards and witches.

Fortunately, dementors always use the same pattern of attack, which simplifies the defense. Specifically, when several dementors spot a potential victim, they all fly directly toward her along straight lines, at their maximal speed. Although dementors can project some fear into their victim from a distance, they cannot cause any real harm until reaching the victim, and the main concern of the attacked magician is to prevent dementors from reaching her.

The standard defense is a counter-dementor spell, called patronus, which acts as a self-guided missile. When a magician casts a patronus spell at a specific dementor, the spell travels toward the selected dementor with constant speed, and eventually hits the dementor. The spell never misses the selected dementor and never hits a different dementor or any other object. If a spell actually passes through a different dementor, it causes no harm and continues its flight toward the intended dementor. Furthermore, the dementor does not notice an approaching patronus, and thus it continues its straight-line flight toward the magician until colliding with the patronus. When the patronus hits the dementor, the monster looses its power for an extended period of time, and it permanently abandons its attack. Thus, if the magician hits all dementors with patronuses, then she wins the fight.

While professional fighting wizards can produce extremely powerful patronuses and scare off an arbitrarily large number of dementors, ordinary magicians are more limited in their defense means. First, an ordinary magician needs 10 seconds to cast a patronus spell, which means that the time between launching two consecutive patronuses is at least 10 seconds. Note that a magician has fine control over the interval between two patronuses, and she may time this delay as necessary; for example, she may wait 11 or 12 seconds before casting the next patronus. Second, when a magician without military training casts a patronus, the spell dissipates in 20 seconds, which means that it needs to hit a dementor within 20 seconds after the launch.

When an ordinary magician faces multiple dementors, she needs to plan the timing of her spells carefully, to make sure that she hits all dementors before any of them reach her. Furthermore, if she cannot fight them off, she should disapparate (that is, teleport away) without casting any patronuses, since she may be too tired for disapparition after launching patronuses. Thus, an attacked magician may need to make complex optimization decisions, and solving the related optimization problem is beyond mental abilities of ordinary magicians. Your task it to write a program that helps to make the right decision; if you solve this problem, wizards and witches will be able to install your program on their magic palmpilots, and use it to plan their actions in critical situations.

Input

The input includes multiple test cases, which describe different dementor attacks; the number of test cases is at most 20. The first line of a test case includes two positive integers between 1 and 100, separated by a single space. The first integer is the speed of patronuses, and the second is the speed of dementors; both speeds are in feet per second. The other lines are initial locations of dementors, one location per line; the number of dementors is between 1 and 10. The description of each location consists of three integers, separated by single spaces, which are its coordinates in feet. The first two coordinates are between −5000 and 5000, and the third is between 10 and 5000, which means that all dementors are initially at least 10 feet above the ground. The last line of a test case is "−1 −1 −1", which does not represent a dementor location. We assume that the attacked magician is always at (0, 0, 0), and that she does not change her location during the fight. The last line of the input, after the last test case, is "−1 −1".

Output

For each test case, the output is a single line. If the magician can hit all dementors before any dementor reaches her, the output is "`fight`"; otherwise, it is "`disapparate`".

Sample input

```
100 100
0 5000 10
5000 0 10
0 −5000 10
−5000 0 10
3000 4000 10
−1 −1 −1
100 100
0 5001 10
5000 0 10
0 −5000 10
−5000 0 10
3000 4000 10
−1 −1 −1
−1 −1
```

Sample output

```
disapparate
fight
```

Problem G: Yule Ball (☺ ☺ ☺ ☺ ☺)

The Christmas celebration at Hogwarts sometimes includes Yule Ball, which is a ballroom dancing event. If a student wants to attend it, he or she has to come with a dancing partner. Since most students do not yet have dating experience, they feel self-conscious around the opposite gender, and they often have difficulty finding a dancing partner. The girls wait until boys invite them, whereas the boys invite only their female friends, since they are too shy to dance with unfamiliar girls. Unfortunately, this strategy limits the attendance of Yule Ball, since many students cannot find partners. Your task is to write a program that helps to improve the attendance; specifically, it should find the matching of boys to their female friends that maximizes the number of couples.

Input

The input includes multiple test cases, which correspond to different social situations; the number of cases is at most 20. A test case includes multiple lines, where each line specifies a boy and all his female friends. The first word in a line is the boy's name, and the other words are the names of his female friends. Each name is a string of lower-case letters, the length of which is between 1 and 10, with single spaces between names. All students have distinct names; different lines in a test case correspond to different boy; and, for each boy, the list of his female friends does not have duplicates. The total number of students in a test case is between 1 and 200; furthermore, we assume that every boy has at least one female friend, and thus every line includes at least two names. Note that a line may include up to 200 names, which means that we do not have the usual 80-character limit on the line length. The last line of a test case contains two periods separated by a single space ("`. .`"), which does not represent a boy. The last line of the input, after the last test case, also contains two periods with a space between them.

Output

For each test case, the output is an integer on a separate line, which represents the maximal possible number of couples such that the boy and girl in each couple are friends.

Sample input

```
harry hermione
ron hermione
. .
harry ginny hermione
ron hermione luna
neville luna
seamus hermione cho
dean cho
. .
. .
```

Sample output

```
1
4
```

Problem H: Age-restricted events (☺ ☺)

Although most educational and social events in Hogwarts are open to all students, a few events have age restrictions. For example, a student has to be at least seventeen years old in order to participate in the Triwazard tournament, and at least sixteen in order to take apparition lessons. When admitting students to such events, the Hogwarts authorities use a device that checks the age of participants. Although this device may appear magical to Hogwarts students, who do not have much experience with computers, it is actually based on simple software, which retrieves a participant's birthday and compares it with today's date. Your task is to implement the related comparison program.

Input

The input includes multiple test cases; each case is on a separate line, and the number of cases is at most 20. A test case includes two dates separated by a single space; the first is a student's date of birth, and the second is today's date. The dates are in the "`mm-dd-yyyy`" format, where "`mm`" is a two-digit month between `01` and `12`, "`dd`" is a two-digit day between `01` and 31, and "`yyyy`" is a four-digit year between `1909` and `2008`. You may assume that the dates are always correct, which means that the input cannot include impossible dates, such as `02/31/2008`. The last input line is "00-00-0000 00-00-0000" which does not represent a test case.

Output

For each test case, the output is either a student's age or the word "`error`". If the student's date of birth is not in the future, the output is the integer that represents the student's age in whole years. Specifically, if today is the student's birthday, then her age is the difference between today's year and her birth year. If today is not her birthday, then the output is her age at her last birthday. Finally, if today's date is earlier than the student's date of birth, then the output is "`error`".

Sample input

```
07-31-1980 03-29-2008
01-01-1909 01-01-2008
02-02-1909 01-01-2008
01-01-2008 01-01-2008
02-02-2008 01-01-2008
00-00-0000 00-00-0000
```

Sample output

```
27
99
98
0
error
```

Problem I: Hogwarts security (☺ ☺)

After evil Lord Voldemort gathered his followers and began attacks against peaceful magicians, the Hogwarts authorities have decided to tighten the school security. As the first step, they have installed a magic device that recognizes people who enter or leave the school, and logs their names. Since advanced magic is often prone to bugs, this device requires thorough testing before its integration into the Hogwarts security system. Your task is to write a program for testing it, which will search for inconsistencies in its logs. Specifically, your program should identify the following inconsistencies:

- A person leaves the school twice, without coming back between the two departures.
- A person enters the school twice, without leaving between the two arrivals.
- The number of people who have left the school, and did not arrive before leaving, is greater than the number of people initially present in the school.

You may assume that all people have different names, and thus the log entries with the same name always refer to the same person.

Input

The input includes multiple test cases, which correspond to different logs; the number of cases is at most 20. The first line of a test case is a positive integer between 1 and 100, which is the initial number of people in the school. The other lines are log entries that represent arrivals and departures, one entry per line. An entry includes two words, separated by a single space; the first word is either "`arrival:`" or "`departure:`", and the second is the related name, which consists of lower-case letters, with no spaces or any other non-alphabetic characters. The length of each name is between 1 and 10 characters, and the number of entries in a log is between 1 and 1,000. The last line of each test case is a single period ("`.`"), which does not represent a log entry. The last line of the input, after the last test case, is "`-1`".

Output

For each test case, the output is a single line. If the log includes any inconsistencies, the output is "`incorrect`"; otherwise, it is "`consistent`".

Sample input

```
2
arrival: harry
arrival: ron
departure: harry
arrival: ginny
.
2
arrival: harry
arrival: ron
departure: ron
arrival: harry
.
2
arrival: ginny
departure: harry
departure: ron
departure: hermione
.
2
departure: harry
departure: harry
.
−1
```

Sample output

```
consistent
incorrect
incorrect
incorrect
```

Problem J: House elves (☺)

The house elves are little creatures that work as domestic servants in wizarding households. Although some witches and wizards feel that this arrangement is unfair to house elves, the elves themselves greatly enjoy their work, and their greatest fear is to lose their servant positions.

The Hogwarts authorities employ a large number of elves, who perform a range of routine tasks. Since elves enjoy the challenge of improving their productivity, they keep various records and analyze them to identify improvement opportunities. In particular, when elves wash dishes, they record the number of dirty dishes and speed of washing them, which allows them to determine the overall time spent for this work. Your task is to write a program that helps them to compute the dishwashing time.

Input

The input includes multiple test cases, which correspond to different logs, where each case consists of two lines; the number of cases is at most 20. The first line of a test case includes two integers, separated by a single space; the first integer is the duration of the log in minutes, which is between 1 and 100, and the second is the washing speed, in dishes per minute, which is between 1 and 50. The second line is a list of integers, separated by single spaces, which specifies how many dirty dishes arrive from the dining hall, minute by minute; the number of these integers is equal to the log duration. Each integer is the number of dishes that arrives within a minute, which is between 0 and 50. The last line of the input is "`0 0`", which does not represent a test case.

Output

The output is a single integer indicating the number of minutes (or part thereof) elves are actively washing dishes.

Sample input

```
3 4
2 0 3
4 5
10 0 0 1
0 0
```

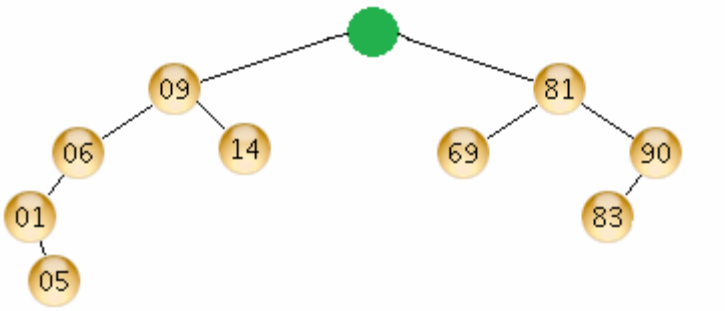
Sample output

```
2
3
```

Problem K: Floo network (☺ ☺ ☺)

A floo network is a magic network of fireplaces, which allows teleportation between different fireplaces. The Hogwarts authorities have recently decided to expand the school floo network, but unfortunately the cost of connecting it with all possible destinations is prohibitive. The authorities need to select among several network plans; for each possible network, they need to determine how often the destinations of potential trips by teachers and students are outside the network.

For example, consider the network in the picture, where the root is Hogwarts, and every other destination is a unique integer between 1 and than 10,000. Suppose that teachers and students want to make twenty trips to the following destinations: 5, 14, 64, 32, 9, 39, 71, 100, 64, 14, 5, 83, 1, 6, 5, 69, 5, 6, 14, 3. Since seven trips involve destinations outside the network, we conclude that the network is ineffective in 7 / 20 = 35% of cases. Your task is to write a program to calculate the percentage of trips that include destinations outside the network.



Input

The input includes multiple test cases; the number of cases is at most 20. Each case begins with two integers, *n* and *m*, separated by a single space, where *n* is between 1 and 10,000, and *m* is between 1 and 1,000,000. The next *n* integers are network destinations, separated by single spaces or line breaks, and the remaining *m* integers are destinations requested by teachers and students, also separated by single spaces or line breaks. Note that this list of integers may span multiple lines, which means that you should count the integers to determine the end of the input. The last line of the input, after the last test case, is "`0 0`".

Output

For each test case, the output is in the form "`Case #k: x%`", where *k* is the case number starting with 1, and *x* is the percentage of requested destinations outside the network, rounded down to the nearest integer.

Sample input

```
9 20
69 83 1 5 6 14 81 90 9
5 14 64 32 9 39 71 100 64 14 5 83 1 6 5 69 5 6 14 3
1 10
5
1 5 2 5 3
5 4 5 5 5
0 0
```

Sample output

```
Case #1: 35%
Case #2: 40%
```

Problem L: Exam preparation (☺ ☺ ☺ ☺ ☺)

When students complete their fifth year at Hogwarts, they demonstrate their magic proficiency in a series of exams. When a student prepares for these exams, she usually has last-minute questions, which require meetings with teachers on the day before the exams. Unfortunately, most teachers are busy before the exams, and their time for meeting with students is limited. When a student requests an appointment, a teacher suggests a time interval when he or she is available, which may overlap with the availability of other teachers. Your task is to write a program that helps a student to make thirty-minute appointments with the maximal number of teachers.

Input

The input includes multiple test cases, which correspond to different lists of teachers and their availabilities; the number of cases is at most 20. A test case consists of multiple lines, where each line includes the name and availability of a specific teacher. The first word in a line is the teacher's name, which is a string of letters followed by a single space; its length is between 1 and 20. The rest of the line is the availability interval, specified by its beginning and end times in the twelve-hour format, including hours and minutes, but not seconds; in particular, `12:00 AM` is the midnight at the start of the day, and `12:00 PM` is the noon. The beginning of each interval is strictly earlier than the end; furthermore, the earliest possible start is `12:00 AM`, and the latest possible end is `11:59 PM`. The number of teachers in a test case is at most 100; different lines represent different teachers, which means that a teacher cannot be available during two different intervals. The last line of a test case is a single period ("`.`"), which does not represent a teacher. The last line of the input, after the last test case, is also a single period.

Output

For each test case, the output is the maximal possible number of thirty-minute appointments with different teachers.

Sample input

```
mcgonagall 10:00 AM 10:30 AM
dumbledore 9:45 AM 10:15 AM
.
flitwick 11:00 AM 2:00 PM
snape 2:34 PM 3:22 PM
hagrid 2:15 PM 3:00 PM
sprout 2:45 PM 3:40 PM
.
.
```

Sample output

```
1
3
```