University Interscholastic League

Computer Science Competition

2001 State Finals Programming Set

I. General Notes

1. Do the problems in any order you like. They do not have to be done in order from 1 to 8.

2. Problems have different point values, depending on their difficulty.

3. Several problems have a reference to columns of an input line, such as column 1 or columns 1-3. In these cases column is referring to the character position on the input line. Column 1 refers to the first character position on the line, while columns 1-3 refer to the first three positions on the line.

4. There is no extraneous input. All input is exactly as specified in the problem. Unless specified by the problem, integer inputs will not have leading zeros.

5. Your program should not print extraneous output. Follow the form exactly as given in the problem.

II. Point Values and Names of Problems

Number	Name	Point Value
Problem 1	Flight Times	4
Problem 2	Que Pasa?	7
Problem 3	What Time Is It?	3
Problem 4	Who turned out the lights?!?!	4
Problem 5	Lost Marbles	5
Problem 6	Who's Calling?	10
Problem 7	Doctor, do I have a fever?	3
Problem 8	Airline Partners	14
Total		**50**

Problem 1	**Flight Times**	4 Points

Program Name: flight.cpp Input File: flight.dat

Flight times in the United States can be estimated using a formula based on the direction of travel and distance between airports. Regardless of the distance or direction, there is also a fixed amount of time for the take-off and landing cycles. Relevant data:

- It takes 0.2 hours for takeoff and climb-out from the origin airport.
- It takes 0.4 hours for approach and landing at the destination airport.
- The aircraft ground speed after climb-out when traveling west is 480 miles/hour.
- The aircraft ground speed after climb-out when traveling east is 540 miles/hour.
- The aircraft ground speed after climb-out when traveling north or south is 510 miles/hour.

Given this data, the flight time can be estimated as:
Flight_time =
0.2 hours for takeoff and climb-out +
0.4 hours for approach and landing +
distance between airports / ground speed

For example, it is 1883 miles from George Bush Intercontinental Airport in Houston, Texas to the Seattle/Tacoma Airport in Seattle, Washington. The travel time (given that the general direction of travel is west) can be computed as

0.2 hours + 0.4 hours + (1883 miles / 480 miles/hour) = 0.6 hours + 3.92 hours = 4.52 hours

Another example is the travel from Dallas/Fort Worth airport to Chicago O'Hare. We will assume that the distance is 1100 miles and the general direction of travel is north. The travel time can be computed as

0.2 hours + 0.4 hours + (1100 miles/ 510 miles/hour) = 0.6 hours + 2.16 hours = 2.76 hours

Input

Input to your program consists of a series of flights each on a line by itself. Each flight's data contains a single flight direction character in column 1. Directions are given as "N" for North, "S" for South, "E" for East, and "W" for West. Column 2 will always be blank. The distance traveled ($0 \le$ distance ≤ 6000 miles) will start in column 3. Distances are expressed in whole miles.

Output

For each flight, your program is to print the approximate flight time to the nearest 1/100 of an hour, starting in column 1.

Example: Input File

```
E 1883
W 1883
N 1100
S 1100
W 983
```

Output to screen

```
4.09
4.52
2.76
2.76
2.65
```

Problem 2	Que Pasa?	7 Points

Program Name: que.cpp **Input File: que.dat**

One of the useful things computers can do is help with the translation of languages. The simplest translators substitute target words of a second language for the original word of a first language. The complex ones try to consider the context of the original words before substituting the target words. In this problem, you will write a simple translator.

Your program will be given a Spanish-to-English dictionary and a series of single-line sentences in Spanish. Your program will parse the Spanish sentence into a sequence of Spanish words and build an English sentence by substituting English words from the dictionary.

For example, consider the following Spanish-to-English dictionary:

```
bonita beautiful
dos two
casa house
mayo may
nosotros we
quiero want
una a
vamos go
yo i
```

Given the following statement in Spanish

```
yo quiero una casa bonita
```

You can use the above dictionary to translate the sentence into English by substituting the English words for the Spanish words:

```
i want a house beautiful
```

Now you can see a major flaw with this approach, but at least you can get the idea of what was said even though the English sentence is not grammatically correct.

Input

Input to your program consists of a Spanish-to-English dictionary followed by a series of Spanish statements to be translated. Line one of the input file consists of a single integer ($1 \le M \le 1000$) which is the number of Spanish/English word pairs in the Spanish-to-English dictionary. The next M lines each consist of 1 Spanish/English word pair. Each word pair consists of the Spanish word starting in column 1 followed by exactly 1 blank followed by the English word. All words are from 1 to 20 characters in length and are lower case. (The letter "n" will be used in the Spanish words in place of "ñ". Also, two "l"'s will be used rather than the Spanish double-"l")

The rest of the file after the Spanish-to-English dictionary will consist of a series of Spanish statements. Each statement will be on a line by itself with the Spanish words separated from each other by a single space. Every Spanish word appears in the Spanish-to-English dictionary and no statement exceeds 80 characters in length, including the blanks between words.

Output

For each Spanish statement, your program should print the equivalent English statements to the screen. Words in each English statement should be separated by exactly one space. No translated statement will exceed 80 characters in length, including the blanks between words.

Example: Input File

```
19
bonita beautiful
pelo hair
dos two
nosotros we
casa house
yo i
todas all
de of
los the
muchachos boys
tienen have
mayo may
hambre hunger
una a
cafe brown
es is
quiero want
vamos go
mi my
yo quiero una casa bonita
todas de los muchachos tienen hambre
mi pelo es cafe
```

Output to screen

```
i want a house beautiful
all of the boys have hunger
my hair is brown
```

Problem 3	What Time Is It?	3 Points

Program Name: miltime.cpp **Input File: miltime.dat**

Personal computers keep their internal time by counting the seconds since an epoch. For example, UNIX frequently uses the epoch of midnight January 1, 1970. In this program, we will instead determine the military time (HH:MM:SS) where the epoch is re-established each day at midnight and an internal clock counts the seconds during a day. Military time means that there is no A.M. or P.M. Instead, the hours of the day are numbered 0 through 23. Zero seconds occurs at midnight and is converted to 00:00:00 in military time and second number 86399 occurs 1 second before midnight (23:59:59 in military time). HH ranges from 0 to 23 and both MM and SS range from 0 to 59.

Input

Input to your program consists of a series of integers ($0 \le S \le 86399$) each on a line by itself and starting in column 1.

Output

For each time in the input file, your program should calculate the corresponding military time of day in the format HH:MM:SS. Your program should print the military time on a line by itself in the format HH:MM:SS (including the colons). If HH, MM, or SS are less than 10, your program must print the leading zero(s) for the affected field(s).

Example: Input file

```
0
86399
873
27342
19842
67829
```

Output to screen

```
00:00:00
23:59:59
00:14:33
07:35:42
05:30:42
18:50:29
```

Problem 4	Who turned out the lights?!?!	4 Points

Program Name: light.cpp Input File: light.dat

There are 3 light switches that control a single light. The light switches can be in the up or down position and are wired such that if exactly 1 of the 3 is in theup position or if all 3 are in the up position, the light is on. The following switch positions would result in the light being on.

1. Up, Down, Down
2. Down, Up, Down
3. Down, Down, Up
4. Up, Up, Up.

All other settings result in a light that is off. Your program will be given an initial set of switch positions followed by a series of switch toggles. Toggling a switch means that if the switch is up, it is changed to down., and if a switch is down, it is changed to up. Your program is to determine the status of the light after all of the switch toggles are applied to the original positions.

Input

Input to your program consists of a series of tests each on a line by itself. Each line consists of the initial switch positions followed by up to 20 switch toggles. Specifically, the line starts with column 1 containing the position of switch 1, column 3 containing the position of switch 2, and column 5 containing the position of switch 3. If the switch is in the up position, the initial position value will be "u". If the switch is in the down position, the initial position value will be a "d". The number of toggles ($0 \le T \le 20$) is given as an integer starting in column 7. The rest of the line contains T switch identifiers (1, 2, or 3) as single digit integers separated by single spaces. These identifiers each represent that the indicated switch was toggled.

Output

For each test, your program should print the state of the light on a line by itself. If the light is in the on state, then your program should print "on". Otherwise, your program should print "off".

Example: Input File

```
u d u 5 1 3 2 3 2
d d d 11 3 2 3 1 3 2 3 2 1 3 2
u u d 14 3 2 1 3 3 2 3 1 1 3 2 1 3 1
```

Output to screen

```
on
on
off
```

Problem 5	**Lost Marbles**	5 Points

Program Name: marbles.cpp **Input File: marbles.dat**

A particular game show has a bonus round in which marbles are dropped in the triangular maze seen in Figure 1. The maze consists of tubes with joints that guide the marble. The 21 joints each contain a deflector that pivots. The positions of the pivots are set based on the contestant's answers to questions and pivot selections. The marble starts at the top of the triangle and, when the marble arrives at a pivot, the pivot's position either deflects the marble to the left or to the right. For example, as the marble arrives at pivot position 8, if pivot 8 is set to the left, the marble will next travel to pivot position 12. The marble will eventually drop out of the maze into one of the bins (A through G) and the contestant will win one of the prize amounts ($100, $500, $1000, or $5000) associated with the bins (as seen in Figure 1).

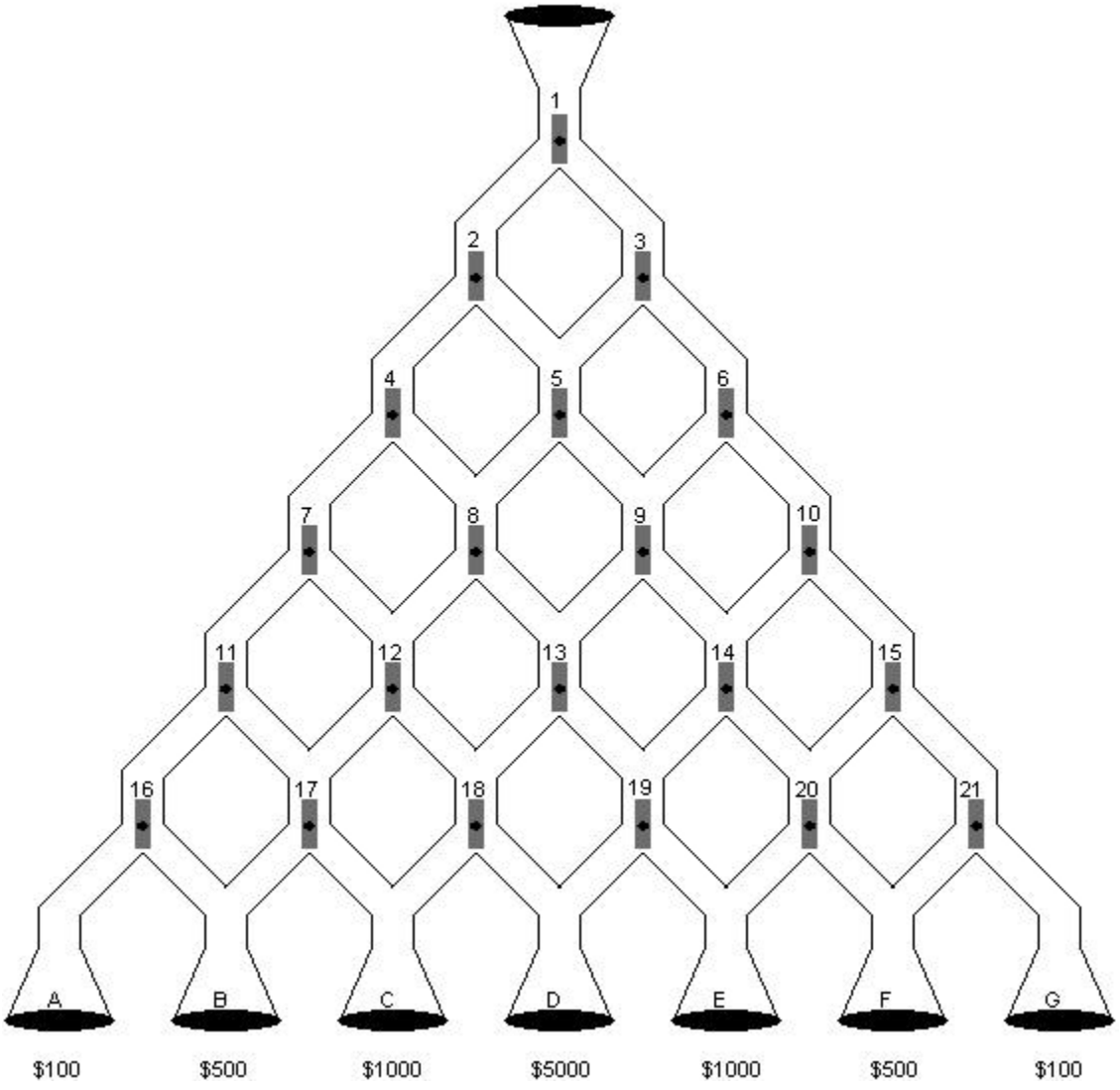

Figure 1: Lost Marbles Maze

Input

Input to your program consists of a series of bonus rounds. Each bonus round is contained on a line by itself with the 21 switch settings (position for switch X appears in column X of the input line) for the round. All switches will

be set either to “L” or “R” for Left and Right respectively. Your program should continue reading bonus rounds until it reaches the end of file.

Output

For each bonus round, your program should output the prize money won with the pivots set according to the input. Your program must print the dollar sign (“$”) and prize amount with no leading, trailing, or embedded spaces starting in column 1. Print each prize amount on a separate line.

Example: Input File

```
LLLLLLLLLLLLLLLLLLLL
LRLRRLLLLRRLLRLRRLRRL
LRRLLRLLRLLLLRRLRRLLR
LRRLLRLLRLLLLRRLLRLLR
```

Output to screen

```
$100
$1000
$1000
$500
```

Problem 6	Who's Calling?	10 Points

Program Name: who.cpp **Input File: who.dat**

Caller ID systems use the media access control (MAC) address (an address that denotes a physical telephone) to search for the caller's name in a database. The data is then sent to the destination telephone for display. Your program is to implement the search of the database using the caller's MAC address. In this program, we will assume that the MAC address is a 5-byte integer given as a series of 5 integers in the range of 0 to 255, inclusive. Your program will be given a customer database that associates MAC addresses to customer names. MAC addresses will appear only once in the database, but there may be many MAC addresses that have the same customer name associated with them (1 person with multiple phones or multiple people with the same name).

Input

Input to your program consists of a customer database followed by a series of calls that require queries of your database. The first line of the input file contains a single integer ($1 \le N \le 100$) starting in column 1 that represents the number of MAC address/customer name entries in the database. Lines 2 through N+1 each contain exactly 1 MAC address/customer name entry. The MAC address is given as a series of 5 integers each between 0 and 255 (inclusive) separated by spaces. The customer name is separated from the 5 integers of the MAC address by a single space and is delimited by forward slashes at the beginning and end. Your program should consider everything between, but not including, the slashes to be the customer name for the MAC address. Customer names will not exceed 30 characters in length (not including the delimiting slashes).
Calls start on line N+2. The MAC addresses for calls are given one per line (5 integers each between 0 and 255 inclusive separated by spaces) and your program should read to the end of file. All MAC addresses of the calls appear in the MAC address/customer name database.

Output

For each call, you program should print the customer name associated with that MAC address. Each customer name should be printed on a line by itself.

Example: Input File

```
10
83 34 134 43 65 /Charlie Brown/
193 82 8 132 224 /Sally Brown/
23 1 45 2 34 /Buck Wheat/
7 35 235 98 2 /Alfalfa Nader/
98 1 83 83 82 /Stimey Rascal/
18 183 28 28 14 /Cotton Rascal/
28 28 91 232 213 /Darla Bea Cutie/
91 38 183 23 91 /Froggie Boy/
182 181 92 255 155 /Dilbert Cartoon/
88 222 222 222 132 /Jane Seymour/
28 28 91 232 213
23 1 45 2 34
182 181 92 255 155
```

Output to screen

```
Darla Bea Cutie
Buck Wheat
Dilbert Cartoon
```

Problem 7	**Doctor, do I have a fever?**	3 Points

Program Name: fever.cpp Input File: fever.dat

Parents are keenly aware of the importance of knowing their infant's weight. All medicine given to an infant is prorated based on the child's weight from some standard weightYou are to write a program that, given a prescription amount (in milliliters), a prescription weight (in pounds) and a child's weight (in pounds), will determine the proper dosage to give to the child.

For example, if the medicine's directions say to give 100 milliliters to a child weighing 48 pounds, then a child weighing 37 pounds should receive a dosage of 77 milliliters. (Your program should always round down to the nearest whole milliliter.) A child weighing 59 pounds should receive a dosage of 79 milliliters for a medicine that prescribes 50 milliliters for a child weighing 37 pounds. Note that it is possible that the dosage is computed (after rounding) to be zero milliliters. In this case, your program will advise the parent to "Consult your child's pediatrician."

A word of caution – if your program simply uses integers throughout the computations, it may be inaccurate due to intermediate computational rounding errors.

Input

Input to your program consists of a series of dosage queries. Each dosage query consists of 3 integers all on a single line. The first integer ($1 \le P \le 1000$) is the number of milliliters in the prescription for a child weighing ($1 \le W \le 100$) pounds as given by the second integer. The third integer ($1 \le C \le 100$) is the weight in pounds of the child needing the dosage your program is to compute.

Output

For each dosage query, your program is to compute the proper dosage (in milliliters) for the child in question. Your program should print the computed dosage on a line by itself with the dosage rounded down to the nearest whole milliliter. If the computed dosage (with rounding) is zero milliliters, your program should not print the dosage and should instead print exactly the following message "Consult your child's pediatrician." on a line by itself. All output should start in column 1 of a new line.

Example: Input File

```
100 48 37
50 37 59
1 1 1
500 50 12
5 50 8
```

Output to screen

```
77
79
1
120
Consult your child's pediatrician.
```

Problem 8	**Airline Partners**	14 Points

Program Name: partners.cpp **Input File: partners.dat**

In order to foster customer loyalty, airlines form alliances and partnerships. Among other benefits, these partnerships allow customers to coordinate collection of frequent flier miles in order to maximize rewards. Customers frequently choose flights on airlines that are partners with the airline that they are collecting miles with. For example, a customer who wanted to fly from Houston to Los Angeles and receive frequent flier miles with Continental Airlines could fly on America West Airlines and still collect miles with Continental.

For the purposes of this program (and to properly test your abilities to write recursive code), a customer will be allowed to also collect frequent flier miles by flying on an airline that is the partner of a partner, or the partner of a partner of a partner, etc. with no limit to the length of a partnership chain. For example, Delta Airlines is a partner of Swiss Air. If Swiss Air is then a partner of Air Egypt, the customer could collect Delta miles by flying on Air Egypt. Your program should consider this to be true even if Delta and Air Egypt do not have a direct partnership with one another. Further, if Air India has a partnership with Air Egypt, then the customer could earn Delta miles on Air India, even if Swiss Air and Delta are not partners of Air India.

Input

Input to your program has threesegments. First, your program is given a list of airlines. The first line of input contains a single integer ($1 \le A \le 100$) indicating the number of airlines to be considered by your program. The next A lines each contain exactly one airline name starting in column 1. Airline names are subject to the following rules:

1. Airline names are between 1 and 20 characters in length.
2. A single underscore ("_") character will be used in place of airlines with spaces in their names.
3. Airline names contain only lower case characters and underscores. For example, Costa Mesa airlines would appear as "costa_mesa".
4. Airline names are listed only once in the first segment.

The second segment of input is a list of partnership agreements. The first line of this segment contains a single integer ($1 \le P \le 100$) representing the number of partnership agreements. The next P lines each contain one partnership definition consisting of exactly two airline names from the list in the first segment. The first partner's name will start in column 1 and will be followed by a single space followed by the second partner's name. The order of the listing in the partnership agreement has no significance, i.e. the partnership is bi-directional. No single partnership agreement line will list the same airline name twice.

The third segment of the input file starts with a single integer ($1 \le Q \le 100$) indicating the number of queries your program is to process. The next Q lines each contain 1 query on a line by itself. Each query consists of a pair of valid airline names with the first starting in column 1 and exactly 1 space separating the two names. No single query will list the same airline name twice.

Output

For each partnership query, if there is a partnership or partnership chain for the two airlines, your program should print the message "PARTNERS" on a line by itself starting in column 1. Otherwise, your program should print the message "No miles for you" starting in column 1.

Example: Input File

```
14
continental
delta
austrian_airways
alaska
america_west
mesa
twa
virgin_atlantic
air_china
southwest
swiss_air
northwest
air_france
air_canada
13
continental america_west
continental air_china
america_west mesa
continental alaska
america_west alaska
continental air_france
america_west twa
continental virgin_atlantic
virgin_atlantic air_france
delta swiss_air
austrian_airways delta
delta austrian_airways
delta air_canada
4
delta america_west
continental mesa
southwest delta
twa air_france
```

Output to screen

```
No miles for you
PARTNERS
No miles for you
PARTNERS
```