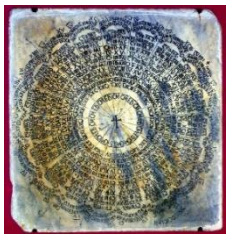
Consortium for Computing Sciences in Colleges
2022 Virtual Southeastern Programming Competition
Saturday, November 12th, 10 AM – 1 PM EST

There are nine (9) problems in this packet. These problems are NOT necessarily sorted by difficulty. You may solve them in any order. Remember input/output for the contest will be from `stdin` to `stdout`. `Stderr` will be ignored. Do not refer to or use external files in your source code. Extra white space at the end of lines is ignored, but extra white space at the beginning or within text on a line is not ignored. Reminders for our virtual competition:

- Teams may be comprised of 1 – 4 students.
- Internet access is allowed.
- Video, audio, or text conferencing between team members is allowed, as is any file transfer.
- **Any team found to be communicating with someone other than their teammates for assistance will be disqualified.**
- Four programming languages are allowed: C#, C++, Java 11, Python 3.8.
- The sample input shown on this sheet has been preloaded into the contest system. The judges will be testing your program on a different, comprehensive set of input based on the written specifications of the problem.
- An awards ceremony will be held at 1:30 PM EST on our Zoom link.
- Have a lot of fun and good luck! ☺

Problem 1

Easter Sunday



Have you ever wondered how the date of Easter is calculated and why it moves to a different Sunday every year? This first problem asks you to construct a computer program that will unveil that mystery. Easter Sunday always falls on the first Sunday after the first full moon of spring. The algorithm below can be used to calculate Easter Sunday between the years 1982 and 2048. Remember that *mod* is short for modulo, and represents the remainder of division.

- Let `a` equal `year` *mod* `19`.
- Let `b` equal `year` *mod* 4.
- Let `c` equal `year` *mod* `7`.
- Let `d` equal `(19 * a + 24)` *mod* `30`.
- Let `e` equal `(2 * b + 4 * c + 6 * d + 5)` *mod* 7.
- Let `f` equal `(22 + d + e)`. If `f` is less than or equal to 31, then the date falls on day `f` in March. Otherwise, you will need to adjust appropriately for an April date.

Input

The first line of input will be a positive integer *n* representing the number of test cases. You may assume that $1 \leq n \leq 100$. This will be followed by *n* lines of input each containing the year in question. If the year falls outside the range from 1982 through 2048, you should output "`Date out of range`" as seen below.

Output Corresponding to Sample Input

The output should be formatted exactly as shown below where each line indicates the date of Easter for the year in question exactly as shown below. Be sure a colon and space follow the year and precede the date as seen. If the year falls outside the range from 1982 up to 2048, you should output "`Date out of range`" as seen below.

Sample Input

```
5
2023
2022
2010
1986
2049
```

Sample Output

```
Easter Sunday in 2023: April 9
Easter Sunday in 2022: April 17
Easter Sunday in 2010: April 4
Easter Sunday in 1986: March 30
Date out of range
```

Problem 2

In the game of Wordle, there is a new secret word every day. A player has six guesses at figuring out the correct secret word. Each guess must be a valid word from a word list of correctly spelled words. After each guess, feedback is given as to whether each letter in your guess is in the right position compared to the secret word.

Your job is to write a variation of this popular game with the following rules:

1) Each guess must be a word in the word list. If a guess does not appear in the word list, it does not count as one of their six guesses.
2) A game always ends on the sixth guess or earlier if they guess the secret word correctly.
3) Feedback is given on each incorrect guess. If a letter in the guess matches the letter in the same position as the secret word, that letter is displayed. If a letter in the guess does not appear anywhere in the secret word, that letter is changed to the asterisk character. If a letter in the guess appears in a different location in the secret word, that letter is displayed with an underscore character. For example, suppose the secret word is `heart`, and you guess the word `hates`, then the feedback would be `h___*` since letters `a`, `t`, and `e` are out of place and letter `s` does not appear in the word `heart`.
4) Be wary of secret words that utilize duplicate letters. For example, suppose the secret word is `queen`. If you guess the word `queue`, the feedback given should be `que*_` since the first `e` is correct, but the second `e` is out of place.
5) All games will be won or lost. Once a player wins, there will not be any extra guesses after the win.

Input

The first line of input will contain a single integer *n*, which represents the number of words in the word list. You may assume that $1 \leq n \leq 1000$. This will be followed by *n* words each on a separate line. Next, a line with an integer *D* for the number of days being played where $1 \leq D \leq 100$. For each day *D*, the secret word for that day appears on a line by itself. This is followed by one or more word guesses, one per line. All guesses consist of five lowercase letters.

Output

For each day, include the day number and word as shown in sample output. This is followed by each word guessed, and feedback given based on rule 3) above. If a guess is not in the word list, it should be printed on a line followed by "`was not in the dictionary`". If a user won, it should be printed along with number of guesses. Use the singular word "`guess`" if they won on only one guess. If they lose, print the message "`Unfortunately, your sixth guess was not correct`." Two blank lines should follow each day. Following all days, a summary of days played, win percentage, the current winning streak, and max winning streak should be printed as shown. The win percentage should be displayed as an unrounded integer.

Sample Input

```
20
crass
cress
cross
error
hates
heart
point
pound
racer
radar
rager
razor
sassy
wants
wheat
where
which
while
whine
would
5
heart
hates
heart
razor
racer
radar
rager
error
errur
razor
sassy
crass
cress
cross
sassy
where
which
while
wants
would
whine
wheat
pound
pound
```

Output Corresponding to Sample Input

```
The wordle for day #1 was heart
Your guess #1 was hates = h___*
Your guess #2 was heart = heart
You WON in 2 guesses


The wordle for day #2 was razor
Your guess #1 was racer = ra**r
Your guess #2 was radar = ra**r
Your guess #3 was rager = ra**r
Your guess #4 was error = *_*or
errur was not in the dictionary
Your guess #5 was razor = razor
You WON in 5 guesses


The wordle for day #3 was sassy
Your guess #1 was crass = **_s_
Your guess #2 was cress = ***s_
Your guess #3 was cross = ***s_
Your guess #4 was sassy = sassy
You WON in 4 guesses


The wordle for day #4 was where
Your guess #1 was which = wh***
Your guess #2 was while = wh**e
Your guess #3 was wants = w****
Your guess #4 was would = w****
Your guess #5 was whine = wh**e
Your guess #6 was wheat = whe**
Unfortunately, your sixth guess was not correct


The wordle for day #5 was pound
Your guess #1 was pound = pound
You WON in 1 guess


Played 5
Win % 80
Current Streak 1
Max Streak 3
```

Problem 3

Ted Lasso

Chelsea always makes her own themed Halloween costumes for her family and this year the theme is Ted Lasso. Chelsea wants to be the soccer ball.

Chelsea is pretty good at math and knows that a soccer ball is a truncated icosahedron. She knows it can be constructed from an icosahedron with the 12 vertices cut off such that one third of each edge is cut off at each of both ends, thereby creating 12 new pentagon faces, and leaving the original 20 triangle faces as regular hexagons. Thus, the length of the edges is one third of that of the original edges.

The radius of the circumscribed sphere of a size 4 soccer ball is 25–26 inches, but Chelsea's soccer ball needs to be much larger to fit her. She needs you to help her computer the total area of both black and white shapes for her soccer ball so she has an idea of how much fabric to buy.

She is experimenting with different sizes and would like you to compute the total area of the black shapes and the total area of the white shapes for a number of sample radii. She has found a helpful formula for you. For edge length a, the radius of the soccer ball is $\frac{a}{4}\sqrt{58 + 18\sqrt{5}}$. Write a program that, given the radius of a soccer ball in inches, computes the area in square yards that will be enough for the white and black pentagons.

Input

The first line of input will contain a single integer N, which will indicate the number of test cases. You may assume that $1 \leq N \leq 100$. Each of the remaining N lines of input will contain a positive floating point value representing the radius of a soccer ball in inches.

Output

For each of N radii input, your program will output the total area of the black shapes in square yards and the total area of the white shapes in square yards. Use the exact format below which includes the case number.

Sample Input

```
3
27.5
62.50
102.75
```

Output Corresponding to Sample Input

```
Case 1: White area: 5 square yards. Black area: 2 square yards.
Case 2: White area: 26 square yards. Black area: 11 square yards.
Case 3: White area: 69 square yards. Black area: 28 square yards.
```

Problem 4

Minerva's Toys

Minerva is a puppy, and LOVES her toys. She has also got quite a lot of them. And because she is a puppy, instead of putting them up, she just leaves them at a random position in the *(x,y)* plane. When she resumes play, she wants to know the best place to start so she can have the most fun.

The way Minerva does this is by finding the two closest toys and standing near one of them. She then proceeds to wreak havoc, as puppies do.

Given a list of the toys' locations, Minerva wants you to find the distance between the closest pair of toys.

Input

The first line of input will contain a single integer *N*, which will indicate the number of test cases to follow. You may assume that $1 \leq N \leq 10$. Each case will start with a line with a single integer *P* where $2 \leq P \leq 1000$. The following *P* lines will contain two floating point numbers indicating a single *(x, y)* coordinate.

Output

For each case, you should print out the distance between the two closest points in the set rounded to three decimal places. For instance, if the closest distance was 1.2345, you should print 1.235.

Sample Input

```
1
4
1.0 2.0
2.0 1.0
1.0 3.0
-1.0 5.0
```

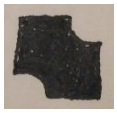
Output Corresponding to Sample Input

```
1.000
```

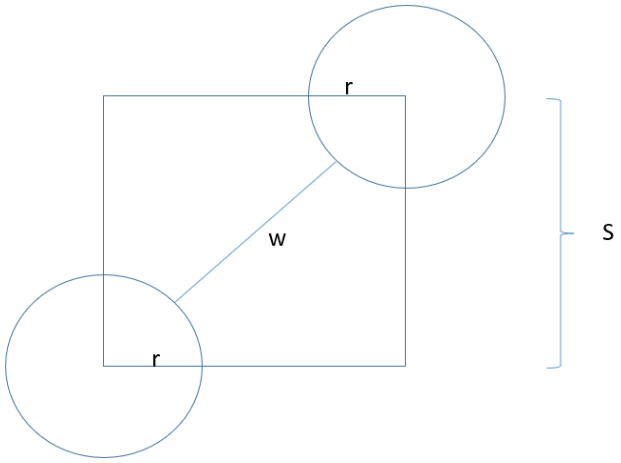
Problem 5

Metal Stamping

Bertrand has started a business that manufactures custom machine parts. One of the parts made at the factory is a thin metal plate. The plate begins as a square ingot of steel. Then, a metal stamping process cuts small quarter circles out of two opposite corners. The resulting shape looks like this.



The metal plates are made in various sizes. Your friend needs some help with their design. Knowing the length of the side of the square ingot that we start with, and the desired finished width of the plate, a quarter circle of what radius needs to be cut from the opposite corners? The following diagram illustrates the problem. We have a square with a side length of *S*. Two circles, each having radius r, are centered at opposite corners of the square. The metal stamper will cut quarter circles out of the square ingot where the two circles intersect with the interior of the square. This will leave a final metal plate shape having a width w. In the diagram, if the line segment of length w were extended in both directions, it would become a diagonal of the square. You need to write a program that can determine the value of *r*, given *S* and *w*.



Input

The first line of the input will contain a positive integer *N*, which is the number of metal plate designs. Each of the next *N* lines of input will contain *S*, the square's side length, followed by *w*, the desired width of the metal plate. The numbers *S* and *w* will be separated by one or more space characters. You may assume that $2 \leq N \leq 100$, and that *S* and *w* are positive real numbers less than 1000. The value of *w* will be less than the diagonal of the square. In every input case, a value for *r* can be determined.

Output

For each test case, there will be one line of output. The format of each line of output is:

`Metal plate <number> needs r = <radius>`

where `<number>` identifies which input case and counts sequentially from 1 up to *N*, and `<radius>` is rounded to the nearest thousandth and formatted to exactly three decimal places.

Sample Input

```
3
1    0.5
2    2
3.5  4
```

Output Corresponding to Sample Input

```
Metal plate 1 needs r = 0.457
Metal plate 2 needs r = 0.414
Metal plate 3 needs r = 0.475
```

Problem 6

Lunch Time

Arielle owns a business that allows its employees to have flexible work schedules. Subject to their supervisor's approval, employees are free to choose when to begin work and when to go home. Their supervisor also allows a lunch break. According to company policy, the lunch break must be given exactly during the middle of an employee's day on the job. Theoretically, every employee in the company could have different work hours and lunch break durations. Therefore, we would like you to write a computer program to determine the start and end times of an employee's lunch break.

An instance of this problem is one employee's work schedule for the day: The time at which the employee is expected to begin work for the day, and when work is finished for the day. In addition, we are given the length of the employee's lunch break, in minutes.

Assumptions:

- All clock times in this program will be expressed using a 24-hour clock, from 0:01 (meaning 12:01 a.m.) to 23:59 (meaning 11:59 p.m.) The duration of the work day and the lunch period are an even number of minutes. The work day will not begin before 0:01 nor end after 23:59.
- The lunch break is to be centered during the work day. In other words, the employee will work for equal amounts of time before and after lunch. For example, if an employee works from 9:00 to 17:00 (9 a.m. to 5 p.m.) and has 80 minutes for lunch, then the lunch period should be 12:20 to 13:40 (12:20 p.m. to 1:40 p.m.)
- The length of the lunch break is less than the total work day. In other words, lunch begins after the work day has started, and lunch ends before the end of the work day. In other words, there will be work to do before and after lunch.

Input

The first line of input will contain a phrase of the form:

`<N> employees`

where N is a positive integer and $2 \leq N \leq 50$. Each of the next N lines of input will have this format:

`start <time> finish <time> lunch break <L> minutes`

where `<time>` represents a time of day on a 24-hour clock, and `<L>` is a positive even integer.

Output

There will be one line of output for each employee. The format of a line of output is:

```
Employee <number> can eat lunch from <time> to <time>.
```

where <number> is a sequential number identifying each employee in sequence from 1 to *N*, and the <time> expressions give the starting and ending times of the lunch break in 24-hour format. The sentence ends with a period. If any time is before 10 o'clock, do not print a leading zero. In other words, 9 o'clock should be printed as 9:00, not as 09:00.

Sample Input

```
5 employees
start 9:00 finish 17:00 lunch break 80 minutes
start 9:00 finish 16:30 lunch break 60 minutes
start 8:30 finish 16:30 lunch break 120 minutes
start 8:00 finish 17:00 lunch break 60 minutes
start 8:00 finish 16:30 lunch break 30 minutes
```

Output Corresponding to Sample Input

```
Employee 1 can eat lunch from 12:20 to 13:40.
Employee 2 can eat lunch from 12:15 to 13:15.
Employee 3 can eat lunch from 11:30 to 13:30.
Employee 4 can eat lunch from 12:00 to 13:00.
Employee 5 can eat lunch from 12:00 to 12:30.
```

Problem 7

Native Alaskan Dice Game

Native Alaskans have a popular dice game called "Animal Names" that helps teach children different names of common animals in two of their twenty Native Alaska languages: Denaakk'e and Gwich'in. The game is a way to add some fun to language learning and pass on the Native Alaskan culture to the next generation by simply rolling three dice. In the game, six animals appear on the faces of the first die: Raven, Eagle, Moose, Caribou, Salmon, and Beaver.

On the second blue die, the names are in Denaakk'e. On the third green die, the animal names are in Gwich'in. The rules of Animal Names stipulate that one player rolls the three dice, but all players try to be the first one to call the roll. There are three different calls: all different, a pair, and all three. The player who first calls the roll correctly wins the point and makes the next roll. The winner is the one with the most total points in a set number of rolls.

Input
The first six lines of input will each consist of three tokens separated by a single space: an animal name, and its equivalent in Denaakk'e and Gwich'in. This will be followed by a line with two integers, *N* and *P*, representing the number of rolls and number of players respectively. You may assume $5 \leq N \leq 100$ and $2 \leq P \leq 5$. This is followed by *N* lines each with four tokens separated by space. The first three tokens represent what is rolled by the three dice in any order. The fourth token represents the initials of which of the *P* players won that particular point. Assume only one game played.

Output
For each of the *N* rolls, your program should print out one line with the initials of the player who won each point along with their call (`all different`, `a pair`, or `all three`) in the format below. This is followed by a blank line, and then a line with the the initials of the player with the most points and their total points as seen below. Assume no ties. You keep playing until a tie is broken.

Sample Input

```
Raven   Dotson  Deetrya
Eagle   Telel   Tthak
Moose   Deneege Dinjik
Caribou Medzeyh Vadzaih
Salmon  Ggaal   LukChoo
Beaver  Noyee   Tsee
16 3
Telel   Eagle   Tthak   AD
Beaver  Tsee    Telel   CH
Tthak   Ggaal   Caribou JK
Salmon  Ggaal   Tsee    AD
Raven   Dotson  Tsee    CH
Noyee   Vadzaih Moose   JK
Dotson  Tthak   Eagle   AD
Beaver  Tsee    Noyee   CH
Deneege Dinjik  Raven   JK
Raven   Dotson  Tsee    AD
Dinjik  Moose   Deneege CH
Medzeyh Beaver  Tsee    JK
Eagle   Tthak   Telel   AD
Ggaal   Tthak   Salmon  CH
Deneege LukChoo Salmon  JK
LukChoo Deneege Salmon  JK
```

Output Corresponding to Sample Input

```
AD won on all three
CH won on a pair
JK won on all different
AD won on a pair
CH won on a pair
JK won on all different
AD won on a pair
CH won on all three
JK won on a pair
AD won on a pair
CH won on all three
JK won on a pair
AD won on all three
CH won on a pair
JK won on a pair
JK won on a pair

JK won the game with 6 points
```

Problem 8

Letter Counts

You are interested in writing a program to analyze the letters used in strings. Specifically, you are interested in which letters appear, which letters do not, and which letter appears the most. The case of the letters does not matter. Assume all letters in the output will be lowercase and that there will always be at least one letter used or missing. All non-letters input are ignored. Your printing of the letters used and missing should appear in alphabetical order. If there is a tie between the letters that appear most often, print the one that would come first alphabetically.

Input

The first line of input will contain a single integer *n*, which represents the number of cases. You may assume that $1 \leq n \leq 100$. This will be followed by *n* test cases. Each test case consists of a string of length *m* where $1 \leq m \leq 80$.

Output

For each test case, print the case number followed by the '>' character as seen below. This is followed by the label "`Used:` ", the letters used in alphabetic order, a semicolon, a blank, the label "`Missing:` ", a semicolon, a blank, the label "`Most Used:` ", and the letter most used.

Sample Input

```
3
The quick brown fox jumped over the lazy dog.
She sold seashells by the seashore.
Go hang a salami...I am a lasagna hog.
```

Output Corresponding to Sample Input

```
Case 1> Used: abcdefghijklmnopqrtuvwxyz; Missing: s; Most Used: e
Case 2> Used: abdehlorsty; Missing: cfgijkmnpquvwxz; Most Used: s
Case 3> Used: aghilmnos; Missing: bcdefjkpqrtuvwxyz; Most Used: a
```

Problem 9

Encryption Matrix

Some modern methods make use of matrices as part of the encryption and decryption process. One way of encrypting a word is to encrypt pairs of letters in the word together. One way of doing this is to fill a 6 x 6 square matrix with the 26 capital letters and the ten digits `'0'` through `'9'`. Each letter and digit appears exactly once in the square. To encrypt a letter pair, the rectangle formed by the two letters is used. Each letter of the original pair is replaced by the letter located in the same row and in the other corner of the rectangle. If both letters happen to be in the same row or same column, the letters are swapped. For example, in the matrix below, `DR` is encrypted as `FP`.

S	T	U	V	W	X
Y	Z	0	1	2	3
4	5	6	7	8	9
A	B	C	D	E	F
G	H	I	J	K	L
M	N	O	P	Q	R

Input

The first six lines of input will be the 6 x 6 square encryption matrix in row major order. This is followed by a single integer *n*, which represents the number of cases. You may assume that $1 \leq n \leq 100$. This will be followed by *n* test cases. Each test case consists of a string of length *m* where $1 \leq m \leq 80$. The string will consist solely of one or more uppercase letters. If the test case contains an odd number of letters, the last letter is unchanged.

Output

For each of the *n* test cases input, print `Word #n` followed by the test case and its encrypted version in the format below.

Sample Input

```
S T U V W X
Y Z 0 1 2 3
4 5 6 7 8 9
A B C D E F
G H I J K L
M N O P Q R
4
DIGITAL
LOGIC
MICROPROCESSOR
COMPUTER
```

Output Corresponding to Sample Input

```
Word #1 DIGITAL is encrypted as CJIGSBL
Word #2 LOGIC is encrypted as IRIGC
Word #3 MICROPROCESSOR is encrypted as OGFOPOORECSSRO
Word #4 COMPUTER is encrypted as OCPMTUFQ
```