Computer Science Competition
Region 2019
Programming Problem 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 12.

2. All problems have a value of 60 points.

3. 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. Unless otherwise specified, your program should read to the end of file.

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

5. A penalty of 5 points will be assessed each time that an incorrect solution is submitted. This penalty will only be assessed if a solution is ultimately judged as correct.

II. Names of Problems

Number	Name
Problem 1	Andy
Problem 2	Belle
Problem 3	Callen
Problem 4	Deborah
Problem 5	Gregory
Problem 6	Jordan
Problem 7	Kavya
Problem 8	Lucie
Problem 9	Mateusz
Problem 10	Nishi
Problem 11	Rosa
Problem 12	Vova

1. Andy

Program Name: Andy.java Input File: None

Andy must write a program to print an ASCII cube. Can you do it before he can? Go!

Input: None.

Output: The pattern exactly as shown in the sample output.

Sample input: None

Sample output:

```
    ooooo+
   ooooo++
  ooooo+++
ooooo++++
xxxxx++++
xxxxx+++
xxxxx++
xxxxx+
```

2. Belle

Program Name: Belle.java Input File: belle.dat

Belle is the manager for the football team at Blooming Prairie High School. Coach Olsen has asked her to find out just the winning score of each of the games the Awesome Blossom football team played in last season. The scores are stored in a file in pairs. There is no indication which team had each score but that doesn't matter because coach just wants to know the winning score. In the event of a tie, print the tie score. Write a program that will help Belle determine the winning score for each game.

Input: A number N representing the number of games, followed by N lines where each line has two scores separated by exactly one space.

Output: The winning (or tie) score for each game printed on a separate line.

Sample input:

```
5
28 21
3 7
0 6
14 24
3 7
```

Sample output:

```
28
7
6
24
7
```

3. Callen

Program Name: Callen.java Input File: callen.dat

Callen was intrigued with the cold weather in the northern US this past winter. He did some research into wind chills and found the following formula:

Formula	Legend
$c = 35.74 + 0.6215t - 35.75s^{0.16} + 0.4275ts^{0.16}$	t = ambient air temp (°F) s = wind speed (mph) c = wind chill (°F)

The ambient air temp must be at or below 50 °F and the wind speed must be at or greater than 3 mph. Callen found Web sites that provide wind chill calculators and display charts but he decides this is a good opportunity to practice his programming skills. However, he feels a little unsure and has asked for your assistance.

Callen wants to be able to display different wind chill charts with specific ranges for both temp and wind speed. For example, starting at 20 °F and ending at −10 °F with a temp step size of 10 while the wind speed starts at 15 mph and ends at 30 mph with a wind speed step size of 5 would produce the following chart:

```
        Wind Speeds
Temps       15      20      25      30
   20      6.2     4.2     2.6     1.3
   10     -6.6    -8.9   -10.7   -12.3
    0    -19.4   -22.0   -24.1   -25.9
  -10    -32.2   -35.1   -37.5   -39.4
```

Input: An unknown number of lines of data but ≤ 10. Each line will contain six integers in the order: $temp_1$, $temp_2$, temp step size, $wind\ speed_1$, $wind\ speed_2$, and wind speed step size. Temps will be between 50 and -120 and wind speeds will be between 3 and 100. Step sizes for both will be between 1 and 25. $Temp_1$ could be either higher or lower than $temp_2$ and wind speed is the same.

Output: A chart precisely formatted as above (and in sample output below) with column widths of exactly 7 characters with all items right-aligned in their columns except for the first row which is left aligned in the second column. The second row displays wind speeds as integers from lowest value to highest value. The first column displays temps as integers from highest value to lowest value. Therefore, an input of −10 20 10 30 15 5 would produce the same result as 20 −10 10 15 30 5 as shown above. Below each table display a line with 25 asterisks "**********".

Sample input:

```
20 -10 10 15 30 5
-110 0 25 70 20 15
```

Sample output:

```
        Wind Speeds
  Temps     15     20     25     30
     20    6.2    4.2    2.6    1.3
     10   -6.6   -8.9  -10.7  -12.3
      0  -19.4  -22.0  -24.1  -25.9
    -10  -32.2  -35.1  -37.5  -39.4
*************************
        Wind Speeds
  Temps     20     35     50     65
      0  -22.0  -27.4  -31.1  -34.0
    -25  -54.8  -61.8  -66.6  -70.4
    -50  -87.6  -96.2 -102.2 -106.7
    -75 -120.4 -130.6 -137.7 -143.1
   -100 -153.2 -165.1 -173.2 -179.5
*************************
```

4. Deborah

Program Name: Deborah.java **Input File: deborah.dat**

Deborah is quite neat in real life, but her computer desktop is a mess! It's cluttered with school assignments, personal photos, and a surprising number of cat videos. After cleanup, Deborah's files are in a nice file tree. A file tree consists of folders and files. Folders can contain files or other folders. To avoid confusion, all of Deborah's files and folders have distinct names.

Deborah has some files or folders that she uses often. These are marked as favorites. Deborah accesses her favorites often. The depth of a file or folder is the number of folders that directly or indirectly contain it. Files and folders on the desktop have a depth of 0.

For example, say Deborah has two folders on her desktop: `catvideos/` and `english/`. These each have a depth of 0. The files and folders inside these folders, like `hamlet.txt` and `tim.mov` each with a depth of 1.

Given a list of Deborah's file tree, with her favorite files and folders marked, what is the largest depth of any favorite file or folder? If multiple favorite files or folders have the same depth, output the one that comes first lexicographically.

Input: The first line of input will contain T, the number of test cases. Each test case starts with N (N <= 50), the number of files and folders on her desktop, followed by N pairs of lines. In the next N sets of data for each test case, each pair of lines describes each file or folder. The first line has the name of the file/folder, if it is a favorite or not (as an integer, 1 means favorite, 0 means not favorite), and the number of items in this file or folder. The number will always be 0 for files. The second line of each pair will have a space separated list of the names of files or folders in the folder. For files, it will be a blank line.

All names will consist only of lowercase English letters and periods and will have length at most 20. No folder will contain itself directly or indirectly. A file or folder will be contained in at most one other folder.

Output: For each test case, output a single line with the name of the favorite file or folder at the greatest depth, as well as that depth value, inside parentheses, formatted as shown below. If there are multiple files or folders as favorites sharing the same depth, output the one that comes first lexicographically.

Sample input:

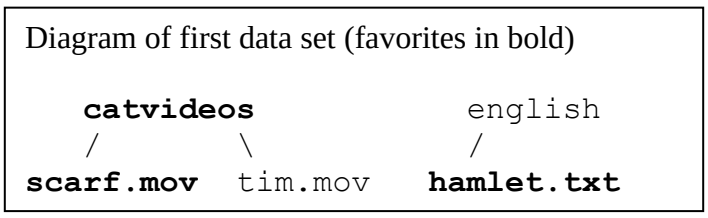
```
2
5
catvideos 1 2
scarf.mov tim.mov
scarf.mov 1 0

tim.mov 0 0

hamlet.txt 1 0

english 0 1
hamlet.txt
5
d 1 1
e
b 1 1
c
c 0 1
d
a 0 1
b
e 0 0
```

Diagram of first data set (favorites in bold)



Sample output:

```
hamlet.txt (1)
d (3)
```

5. Gregory

Program Name: Gregory.java **Input File: gregory.dat**

Gregory is fascinated with Web sites that display a password strength for new passwords. He has developed a new scoring algorithm of his own and needs help to code a solution. To help with this process, he found a designation of the occurrence frequency of letters in the English Oxford Living Dictionaries (en.oxforddictionaries.com), indicated by the alphabet string **`EARIOTNSLCUDPMHGBFYWKVXZJQ`**. This designation declares the letter E as the most often occurring letter in English words, A next, and Q the least often occurring letter.

The scoring rules of his process are as follows:

- Match any of the top 100 common passwords, regardless of letter case, results in score of 0
- Does not contain at least 3 of 4 character categories: lowercase letters, uppercase letters, digits, or special characters, results in score of 0
- Length less than 8 characters results in score of 0
- Passwords not excluded by the above criteria are scored based on content
- Letters, regardless of case, found in the **first** half of the highest-to-lowest frequency list earn 1 point each
- Letters, regardless of case, found in the **last** half of the highest-to-lowest frequency list earn 2 points each
- Digits earn 2 points each
- Special characters earn 3 points each
- Lengths greater than 8 characters earn +2 bonus points for each extra character
- Using all 4 categories (uppercase, lowercase, digits, special characters) earns +5 bonus points
- Category change between sequential characters (e.g., letter to digit, uppercase to lowercase, etc.) earns +2 bonus points for each change. "`UiL`" as part of a password would earn +2 bonus points each for "`Ui`" and "`iL`" case changes and "`u-n`" would earn +2 bonus points each for "`u-`" and "`-n`".
- Immediately repeated characters, of same case for letters, earn a −1 point penalty for each repetition. "`sss`" as part of a password would earn 3 points for the individual letters but get penalized −2 points each for the second and third '`s`'.
- A sequence of 3 consecutive letters of the same case or digits (e.g., "`abc`", "`DEF`", "`789`", etc.) earn −5 penalty points for each sequence

Examples:

- "`password`" and "`Qwerty123`" score 0 since they are in list of most common passwords
- "`pass1234`" scores 0 since it contains only 2 categories of characters: lowercase and digits
- "`fgh123$`" scores 0 since it is less than 8 characters
- "`0circleS`" scores 13 points total: 9 points for characters with +4 bonus points for 2 category changes "`0c`" and "`eS`"
- "`UiL+2019`" scores 27 points total: 14 points for characters with +13 bonus points for using characters from all 4 categories and 4 category changes "`Ui`", "`iL`", "`L+`", and "`+2`"
- "`u20-IL_19`" scores 34 points total: 17 points for characters with +17 bonus points for 2 extra characters and using characters from all four categories, and for 5 category changes "`u2`", "`0-`", "`-I`", "`L_`", and "`_1`"
- "`Sssssss0`" scores 8 points total: 9 points for characters with +4 bonus points for 2 category changes and −5 penalty points for 5 repeated lowercase '`s`' after the initial lowercase '`s`'
- "`N000000s`" scores 13 points total: 14 points for characters with +4 bonus points for 2 category changes and −5 penalty points for 5 repeated '`0`' after the initial '`0`'
- "`Pass12345`!" scores 16 points total: 17 points for characters with +15 bonus points for using characters from all 4 categories, 3 category changes, and 2 extra characters; with −16 penalty points, −1 for repeated '`s`' and −5 points each for "`123`", "`234`", and "`345`"
- "`Abcdefg0`" score -4 points total: 12 points for characters with +4 bonus points for 2 category changes and -20 penalty points for "`bcd`", "`cde`", "`def`", and "`efg`"

(continued next page)

Gregory (cont.)
Strength ratings based on total scores:

- 0 points and lower → UNACCEPTABLE
- 1 – 15 points → WEAK
- 16 – 30 points → FAIR
- 31 – 45 points → GOOD
- 41 points and higher → STRONG

Input: First line contains the letters A…Z, with no spacing and in uppercase, in highest-to-lowest frequency of occurrence according to the English Oxford Living Dictionaries (en.oxforddictionaries.com). Second line contains allowable special characters. The next ten lines contain the 100 mostly commonly used passwords for 2018 in decreasing order (teamsid.com). There are ten passwords per line separated by commas. The remaining lines contain one password per line to be rated for strength. Passwords will not contain spaces or commas and will not exceed 20 characters in length.

Output: For each password, one line of output with the password followed by a colon ':' followed by the calculated score and another colon ':' followed by the strength rating. There must be no additional spacing.

Sample input:
```
EARIOTNSLCUDPMHGBFYWKVXZJQ
`~!@#$%^&*()-_=+[{]}\|;:'"<.>/?
123456,password,123456789,12345678,12345,111111,1234567,sunshine,qwerty,iloveyou
princess,admin,welcome,666666,abc123,football,123123,monkey,654321,!@#$%^&*
charlie,aa123456,donald,password1,qwerty123,zxcvbnm,121212,bailey,freedom,shadow
passw0rd,baseball,buster,daniel,hannah,thomas,summer,george,harley,222222
jessica,ginger,letmein,abcdef,solo,jordan,55555,tigger,joshua,pepper
sophie,1234,robert,matthew,12341234,andrew,lakers,andrea,1qaz2wsx,starwars
ferrari,cheese,computer,corvette,mercedes,blahblah,maverick,hello,nicole,hunter
1989,amanda,1990,jennifer,banana,chelsea,ranger,1991,trustno1,merlin
cookie,ashley,bandit,killer,aaaaaa,1q2w3e,zaq1zaq1,test,hockey,dallas
whatever,admin123, asdf-123,liverpool,querty,william,soccer,london,1992,biteme
password
Qwerty123
pass1234
fgh123$
0circleS
UiL+2019
u20-IL_19
Sssssss0
N000000s
Pass12345!
Abcdefg0
jfm2amj0^jas1ond9
```

Sample output:
```
password:0:UNACCEPTABLE
Qwerty123:0:UNACCEPTABLE
pass1234:0:UNACCEPTABLE
fgh123$:0:UNACCEPTABLE
0circleS:13:WEAK
UiL+2019:27:FAIR
u20-IL_19:34:GOOD
Sssssss0:8:WEAK
N000000s:13:WEAK
Pass12345!:16:FAIR
Abcdefg0:-4:UNACCEPTABLE
jfm2amj0^jas1ond9:63:STRONG
```

6. Jordan

Program Name: Jordan.java **Input File: jordan.dat**

Jordan just learned about Fibonacci numbers and how it is the worst type of problem to do with recursion. The Fibonacci sequence is where the next number is the sum of the previous two, beginning with 1 and 1, resulting in the sequence 1, 1, 2, 3, 5, 8, 13, 21, and so on. He has decided to write a problem with Fibonacci, since it is so common, but decided to add a twist. Instead of just doing a simple Fibonacci pattern, he decided to do the whole thing in base 17. The standard Fibonacci sequence in base 17 would be:

1, 1, 2, 3, 5, 8, D(13_{10}), 14(21_{10}), and so on.

For the value 8 in the sequence, the index position is 6 since it is the sixth value in the sequence.
The index of 14_{17} is 8.

To make it a bit more interesting, he also decided to give you a beginning two-number sequence other than the traditional 1 and 1, and then wants to know if a third target value is in the resulting base 17 Fibonacci-style version of the sequence. For example, if the first two values of the sequence were 4 and A, the Fibonacci-style base 17 sequence would be:

4, A(10_{10}), E(14_{10}), 17(24_{10}), 24(38_{10}), 3B(62_{10}), and so on.

If the target value is 3B, clearly it is in the sixth position of the sequence. If the target value is 20_{17}, it is not found in the sequence, but falls between 17_{17} and 24_{17}.

Input: The first line of the data file contains a count of the number of data sets. Each dataset will have three base 17 values. The first two represent the first two numbers in the base 17 Fibonacci sequence, the third the target. All base 17 numbers will be < $33D3D8307B214008_{17}$. The third number will always be greater than or equal to the second number in the sequence.

Output: Print either the base 10 index of the target number, or if the target is not in the sequence, the base 17 values in the sequence immediately before and after the target. All alpha characters in the base 17 values must be uppercased.

Example Input:

```
4
4 A 3B
4 A FF
1 3 12B3
3 B B94G3
```

Example Output:

```
6
F7 17G
C61 12GF
26
```

7. Kavya

Program Name: Kavya.java Input File: kavya.dat

Kavya has a fascination with sorts and letters. He has now come up with a process that sorts letters in a sentence depending on the frequency of letter occurrence. He also thinks all letters should be uppercase but keeps forgetting to hit the shift lock to enter all his letters in uppercase, but I'm sure you can help him with that issue. He doesn't want to sort anything else besides actual letters, which means for you if it's not a letter you get to ignore it and not include it in the final sorting.

For example, take the sentence:
"Yes, we finally made it to Regionals!"

Using the sorting routine described above, the result would be:
EEEEIIIAAALLLYYSSNNTTOOWFMDRG

Clearly the letter E has the greatest frequency (occurs more often than any other letter), and so all the Es are listed first. The letters I, A and L each occur three times, listed in the order they each first occur, the "I" in the word "finally", the "A" two letters later in the same word, with "L" immediately after that. "Y", "S", "T", "N", and "O" each occur twice and are listed in the order they first occur in the same manner, followed by the last six letters, each occurring only once.

Input: Several sentences each on its own line.

Output: All the letters in the sentence sorted by the number of times the letter occurs in the sentence, as described and demonstrated in the example above. If multiple letters have the same frequency of occurrence, then sort by when the letter first occurred in the sentence.

Sample Input:
```
Yes, we finally made it to Regionals!
Why do you think sorting seems to be so important in Computer Science?
This was not as hard of a problem as it seems to be.
```

Sample Output:
```
EEEEIIIAAALLLYYSSNNTTOOWFMDRG
OOOOOOOTTTTTTEEEEEEIIIIINNNNNSSSSSRRRMMMCCCHHYYUUPPWDKGBA
SSSSSSAAAAATTTTOOOOEEEEHHIIRRBBMMWNDFPL
```

8. Lucie

Program Name: Lucie.java Input File: lucie.dat

The UIL Computer Science team needs your help! They have contest program scores stored where one row of data contains each of the team scores for a single program as shown below:

	Team 1	Team 2	Team 3	Team 4
Program 1	60	55	60	60
Program 2	55	45	0	60
Program 3	0	50	55	60

The UIL Computer Science team needs the data in the following form:

	Program 1	Program 2	Program 3
Team 1	60	55	0
Team 2	55	45	50
Team 3	60	0	55
Team 4	60	60	60

The above process is known as transposing rows and columns; rows become columns and columns become rows. In addition to transposing the raw data, the UIL team has also asked to get row and column totals as shown below:

	Program 1	Program 2	Program 3	Total Score
Team 1	60	55	0	115
Team 2	55	45	50	150
Team 3	60	0	55	115
Team 4	60	60	60	180
Totals	235	160	165	

Input: The first line of the data file contains one positive integer, the number of contests or data sets $S \leq 20$. Each set of data then contains one line with two positive integers separated by a comma, the number of teams T and the number of programs P. The input continues with T lines of P integer scores separated by commas. The largest contest has a maximum of 50 teams with a maximum of 12 programs. Scores range from 0 to 60.

Output: For each contest, output a list of values for each team with a total score as the last item on the line. Then output one line containing the totals for each program. All output lines must separate the values by exactly one comma. Following the data for each contest, display a line containing exactly 12 asterisks "**********".

Sample input:

```
2
3,4
60,55,60,60
55,45,0,60
0,50,55,60
5,3
50,60,55
60,0,50
0,50,60
55,45,60
50,60,0
```

Sample output:

```
60,55,0,115
55,45,50,150
60,0,55,115
60,60,60,180
235,160,165
**********
50,60,0,55,50,215
60,0,50,45,60,215
55,50,60,60,0,225
165,110,110,160,110
**********
```

9. Mateusz

Program Name: Mateusz.java **Input File: mateusz.dat**

Gennady's little brother Mateusz received a set of stamps for Christmas, all of which are perfect rectangles. He likes to use multiple rectangles in order to create art. Gennady, always the analyst, is more interested in the area covered by at least one stamp. Given a description of Mateusz's artwork, can you answer his question?

Input: The data file will begin with T (T <= 10), the number of distinct artworks Mateusz has created. The first artwork is Work #1, the second is Work #2, and so on.

For each artwork, the first line of input has N (N <= 12), the number of stamps Mateusz used in this piece. The following N lines each contain 4 integers, x_i, y_i, w_i, and h_i. This means that there is a rectangle with bottom left corner at (x_i, y_i), and top right corner at (x_i + w_i, y_i + h_i). It is guaranteed that every point within a given rectangle is in the range [-10^8, 10^8]. All of Mateusz' artworks are on different sheets of paper.

Output: For each work, output a single line: the total area covered by some stamp, formatted in the same manner as the samples.

Sample input:

```
3
2
0 0 3 3
2 2 3 3
1
0 0 1 1
2
0 0 1000000 1000000
100 100 1 1
```

Sample output:

```
Case #1: The total area is 17.
Case #2: The total area is 1.
Case #3: The total area is 1000000000000.
```

10. Nishi

Program Name: Nishi.java Input File: nishi.dat

Nishi has just learned all about significant digits in chemistry class. Significant digits are a way to indicate certainty of measurement when reporting the results of various calculations. Significant digits are defined by the following rules (significant digits for the examples are shown in parenthesis):

- All non-zero digits are significant: 1, 2, 3, 4, 5, 6, 7, 8, 9.
- Zeros between non-zero digits are significant: 102 (3), 2005 (4), 50009 (5).
- Leading zeros are never significant: 0.02 (1), 0.000515 (3).
- In a number with a decimal point, trailing zeros (those to the right of the last non-zero digit) are significant: 2.02000 (6), 5.400 (4), 57.5400 (6).
- In a number without a decimal point, trailing zeros may or may not be significant. To keep things simple, Nishi's teacher has told them to assume they are NOT significant.

When multiplying and dividing, results should have as many significant figures as the number with the least number of significant digits. For example, $1.234 \times 2.0 = 2.468$. However, since 2.0 only has two significant digits, the answer should be reported as 2.5.

To round properly to the correct number of significant figures, ONLY consider the digit to the right of the last significant digit. For example, 5.349 rounded to two significant figures would be 5.3. The digit 9 is not considered in the rounding process. If the digit considered for rounding is 5, round towards positive infinity.

While Nishi is to consider trailing zeroes in numbers with no decimal place in the problems as not significant this can't be the case for numbers in the results. To remove any doubt about the number of significant digits in results that turn out to be whole numbers with trailing zeroes Nishi is going to display all of the results using scientific notation. For example 18000 with three significant digits should be printed as 1.80E4. That is the equivalent of $1.80 \text{ X } 10^4$. If a result has just one significant digit, do not display a decimal point. For example, 19825 with one significant digit would be 2E4.

Nishi's homework is a worksheet with a variety of multiplication and division problems for her to do where the answers are to be rounded to the correct number of significant digits. Nishi has wisely enrolled in a computer science class as well as chemistry this school year so she is going to code up a program that will do the calculations in the worksheet and report the answers in the correct form for her. What might that program look like?

Input: A number N on the first line that represents the number of problems on the worksheet. N lines where each line is a problem in the form A X B or A / B where X stands for multiplication and / stands for division. A and B will both be greater than -100,000 and less than 100,000. There will be exactly one space between the operands and the operator.

Output: The product or quotient for each problem, each on a separate line. Each value should be displayed using scientific notation and with the proper number of significant digits.

Sample input:	**Sample output:**
4 3.566 X 1.5 30.0 X 8.973 140 / 19.2654 1500.0600 X 35000.00005	5.3E0 2.69E2 7.3E0 5.2502100E7

11. Rosa

Program Name: Rosa.java Input File: rosa.dat

Rosa called last night and needs a program that takes two sets of data and compares an item in both sets of data. The data will have three fields separated by a comma. Each field could be a key. Finally, the key she wants to compare could change each time. The first line of data is the header information.

When comparing the data, she really doesn't tell you what she wants, so you decide you will give her three items of information: what is missing from the second file but is contained in the first file, what is missing from the first file but is contained in the second file, and what is different between the two files.

The key she gives you is guaranteed to be unique in the data.

Input: The first line is the number of data sets. This will be followed by the key Rosa wants to use to compare. The next line will be the name of the file. The following line will be three possible keys separated by commas (Note: Both files' keys will always be in the same order). This will then be followed by an unknown amount of data and will end with "EOF". After EOF the next file will be entered, following same format as first file (file name, keys separated by commas, then data).

Output: Will have three basic items for each data set, as described below. Note **bolded text** represents each file name, and *italics* represent information. `Normal text` is exact.

`First_File_Name` `Missing from` **`Second_File_Name`**
All data contained in first file, but not second file, sorted by first file key
`Second_File_Name` `Missing from` **`First_File_Name`**
All data contained in second file, but not first file, sorted by second file key
`Difference between Files`
All data different between the files, sorted by key first, then information from first file, then information from second file.

(continued next page)

Rosa (continued)

Sample Input

```
2
One
FILE 1
One,two,three
4,bob,can
7,house,short
8,goof,short
EOF
FILE 2
One,two,three
4,bob,can
7,course,crank
9,hobbit,googball
EOF
H2
FILE A
H1,H2,H3
Airhead,A,Can
Dog,D,Here
UIL,U,Bob
EOF
FILE B
H1,H2,H3
Dog,D,Here
Airhead,A,Hook
UIL,U,Canister
Wha,W,Waycar
Habbit,H,Honolulu
EOF
```

Sample Output

```
File 1 Missing from File 2
8,goof,short
File 2 Missing from File 1
9,hobbit,googball
Difference between Files
7,house,short
7,course,crank
File A Missing from File B
File B Missing from File A
Habbit,H,Honolulu
Wha,W,Waycar
Difference between Files
Airhead,A,Can
Airhead,A,Hook
UIL,U,Bob
UIL,U,Canister
```

12. Vova

Program Name: Vova.java Input File: vova.dat

Like most UIL problems, Vova is a person. Just kidding! Vova is a snake! Surprise! Hisssss.

Vova exists on an infinite cartesian plane. His home is at (x1, y1). Kaepora the wise owl told Vova about a tasty mouse that lives at (x2, y2). Vova is a hungry snake and wants to eat this mouse. However, he also doesn't want to leave his home unguarded, so he leaves his tail at home. Vova has a length of L and can move one unit along the x or y axis, but not both. For example, if his head is at (3, 4), Vova can move to (3, 5), (3, 3), (4, 4) or (2, 4).

Before hunting, Vova is curled up in his hole, but to digest the mouse, he needs to be fully stretched out. This means his path from his hole to the mouse must be of length exactly L. However, Vova is quite hungry, and if he comes across a spot where his body is laying, he might eat it! Therefore his path cannot self-intersect.

Formally, find a sequence of points of length L + 1 such that

- the first element of the sequence is (x1, y1)
- the last element of the sequence is (x2, y2)
- The distance between consecutive elements of the sequence is 1.
- No point appears more than once in the sequence

Input: The first line of input is T (T <= 50), the number of test cases. Each of the next T lines has one test case with 5 numbers on it: x1, y1, x2, y2, L. (x1, y1) is the position of Vova's hole, (x2, y2) is the position of the mouse, and L is the length of Vova the snake.

-10^8 <= x1, y1, x2, y2 <= 10^8
1 <= L <= 10^16

Output: For each test case, output "YES" if there is some path where Vova can eat the mouse without self intersecting and "NO" otherwise. Format as in the samples.

Sample input:

```
3
1 2 7 6 12
2 0 -2 0 3
-1 -1 1 1 4
```

Sample output:

```
Case #1: YES
Case #2: NO
Case #3: YES
```