University of Texas at Austin - UIL Computer Science Contest

General Notes:

1. Do the problems in any order you like. The problems do not have to be completed in order from 1 to 12.

2. For problems that read from a data file: When the program is judged, the data file will be in the same directory as your program. Do not include any extraneous path information in your program with the data file name.

3. Submit your source code, the .java file, for judging via the PC^2 software.

4. All problems are worth 60 points. Incorrect submissions receive a deduction of 5 points, but **ONLY** if the problem is ultimately judged correct. Problems may be reworked and resubmitted as many times as you like.

5. There is no extraneous input. All input is exactly as specified in the problem. Integer input will not have leading zeros unless specified by the problem.

6. Your program shall not print extraneous output. Follow the format exactly as given in the problem statement and as shown in example output.

7. The time limit for problems is 120 seconds. If the program has not terminated after 120 seconds on the judge's computer, the program will be judged as an incorrect submission.

Number	Problem Name
1	CardSet
2	Change
3	Freshener
4	Happy
5	Hint
6	Population
7	QBRating
8	Rare
9	Ride
10	Slugging
11	Soccer
12	Tower

1. Game, Set, Match

Program Name: CardSet.java **Input File: cardset.dat**

Set is a card game created by Marsha Jean Falco, a geneticist. Write a program to determine if three cards form a *set.* Cards in the game of set have four features:

- a type of symbol (diamond, squiggle, or oval)
- a number of symbols (1, 2, or 3)
- a shading of the symbols (solid, striped, or open)
- a color (red, green, or purple)

Three cards form a set if they meet all four of the following criteria:

- all three cards have the same type of symbol or each card has a different type of symbol
- all three cards have the same number of symbols or each card has a different number of symbols
- all three cards have the same shading of the symbols or each card has a different shading of the symbols
- all three have the same color or each card has a different color

For this question, each card is represented by a series of characters.

- diamonds are represented by the `^` character, squiggles are represented by the `~` character, and ovals are represented by the `0` (zero) character
- the number of symbols on a card will vary from 1 to 3 (For example `^`, `^^`, or `^^^`. Each card will only have one type of symbol, but the number of symbols will vary from 1 to 3.)
- the card's shading is represented by a separate character, `S` for solid, `T` for striped, and `N` for open
- the card's color is represented by a separate character, `R` for red, `G` for green, and `P` for purple

So for example, a card with 3 solid red diamonds is shown as: `^^^SR` , a card with 2 striped red squiggles is shown as `~~TR` , and a card with 1 open red oval is shown as `0NR` . Those three cards form a set because they have different symbols, different numbers, different shading, and the same color.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will consist of exactly 3 lines. Each line represents a single card as described above.

Output

For each data set print out `SET` if the three cards form a set per the rules of the game or `NOT` if the three cards do not form a set.

Example Input File

```
5
^^^SP
000TP
~~~NP
^^^SR
~~TR
0NR
^^^SP
000SB
~~TP
~NG
~~TR
~~~NG
~SR
~SR
~SR
```

Example Output To Screen

```
SET
SET
NOT
NOT
SET
```

2. Mom, Send Money

Program Name: Change.java **Input File: change.dat**

College students are often hurting for money. They have to search drawers for spare change and bills to afford Ramen noodles for the week. Write a program that given the amount of each coin and bill, prints out the total dollars truncated to the nearest whole dollar.

The denominations for this problem are:

Penny, 1/100th of a dollar
Nickel, 1/20th of a dollar
Dime, 1/10th of a dollar
Quarter, 1/4th of a dollar
Half Dollar, 1/2th of a dollar
Dollar Coin, 1 dollar
Dollar Bill, 1 dollar
Two Dollar Bill, 2 dollars
Five Dollar Bill, 5 dollars
Ten Dollar Bill, 10 dollars
Twenty Dollar Bill, 20 dollars
Fifty Dollar Bill, 50 dollars
Hundred Dollar Bill, 100 dollars

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will be a single line with 13 integers separated by a single space.
- Each integer in a data set will be greater than or equal to zero.
- The first integer in a data set indicates the number of pennies, the second integer represents the number of nickels, and so forth. The last integer in the data set represents the number of hundred dollar bills.

Output

For each data set output the total number of dollars truncated to the nearest whole dollar. (In other words print the number of whole dollars in each data set, but leave off any fractions of dollars.)

Example Input File

```
5
4 0 2 3 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0
1 2 3 4 5 6 7 8 9 10 11 12 13
4 0 2 3 10 10 2 1 2 1 10 1 100
10 10 10 10 5 5 5 5 2 2 2 2 1
```

Example Output To Screen

```
0
0
2297
10289
296
```

3. Smells Like Teen Spirit

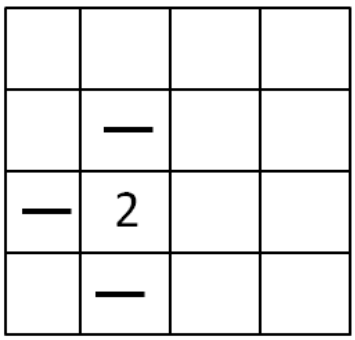
Program Name: Freshener.java Input File: freshener.dat

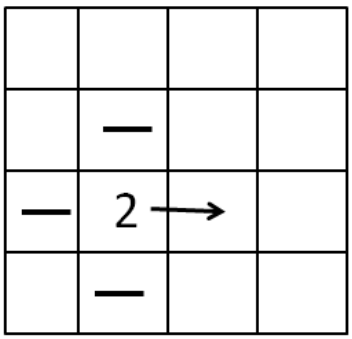
Write a program to determine how many spaces in a room are covered by more than one air freshener and how many spaces are not covered by any air fresheners.

The room layout will be a rectangular matrix of characters. The allowed characters are `*`, `-`, `|`, and the digits `1` through `9`. `*`'s represent open spaces. `-`'s and `|`'s represent walls. Digits `0` through `9` represent open spaces with an air freshener in the space.

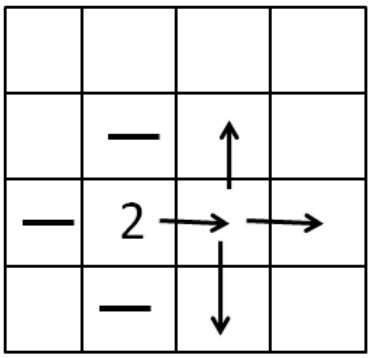
The digit for an air freshener indicates the range of that air freshener: the number of spaces the fragrance from the air freshener travels. The fragrance is stopped by the boundaries of the room and walls. The fragrance does not "rebound" off boundaries or walls. A range of 0 means the fragrance does not travel from the space that holds the air freshener. Each time the fragrance moves to a new space its range is reduced by 1. The fragrance from an air freshener spreads to adjacent spaces (up, down, left and right only) unless those spaces are walls or boundaries. When a fragrance reaches a new space, it spreads to spaces adjacent (up, down, left, and right only) from the new cell. This continues until the number of spaces the fragrance has spread from the original space equals the range of the air freshener. The fragrance does not travel

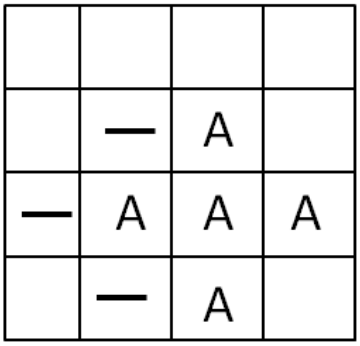
Consider this example, with a single air freshener with a range of 2. Open spaces are shown with blanks. There are walls above, below, and to the left of the air freshener, so initially the fragrance can only move to the right one space:





After moving to the right one space the fragrance's range is reduced by 1. The fragrance may now move from the cell to the right of the 2, up, down, and right another space. The spaces the fragrance from the air freshener ultimately reach are shown with capital A's.





The building layout is rectangular. A layout with 6 rows and 4 columns per row could look like this:

```
3*|*
*-|*
****
*-|*
2**0
*|**
```

Consider the spread of each air fresheners individually (A for the 3, B for the 2 and C for the 0) gives the following spreads of fragrance:

```
AA|*      **|*      **|*
A-|*      *-|*      *-|*
AA**      B***      ****
A-|*      B-|*      *-|*
****      BBB*      ***C
*|**      B|**      *|**
```

With the overlap of fragrances from the air fresheners there are 2 spaces in the layout covered by two or more air fresheners and 7 spaces (not counting walls) not covered by any air freshener.

Input

- The first line will be a single integer N that indicates the number of data sets.
- The first line of each data set will contain two integers R and C separated by a single space. R is the number of rows and C is the number of columns for this data set.
- The next R lines in the data set will contain C characters.
- All characters in the room layout will be one of these: `-|*0123456789`

Output

For each data set output two integers M and Z where M is the number of spaces in the room covered by two or more air fresheners and Z is the number of spaces in the room not covered by any air fresheners. Walls are not included in either total.

Example Input File

```
5
6 4
3*|*
*-|*
****
*-|*
2**0
*|**
1 17
9******101******8
8 10
**|***3*|*
**3***3*|*
**|0**--|*
*7|1**|*|*
**|**--*|*
**|*****|*
**|**---|*
-----***98
7 7
*******
*******
*******
***5***
*******
*******
*******
6 7
9|*6*|9
*|*|*|*
*|*|*|*
*|*|*|*
*|*|*|*
0**|***
```

Example Output To Screen

```
2 7
5 0
36 10
0 4
7 0
```

4. :(Numbers, :) Numbers

Program Name: Happy.java **Input File: NONE**

Consider the following algorithm on a positive integer: take all of the individual digits of the number, square each, and add the resulting integers together. For example:

- given the positive integer 200, its digits are 2, 0, and 0
- squaring the digits results in 4, 0, and 0
- adding the squares together gives 4 + 0 + 0 = 4

Repeat the process until the result is 1 or a cycle occurs.

Continuing the above example: (number -> digits -> squares -> sum of squares)

- `4 -> 4 -> 16 -> 16`
- `16 -> [1, 6] -> [1, 36] -> 1 + 36 = 37`
- `37 -> [3, 7] -> [9, 49] -> 9 + 49 = 58`
- `58 -> [5, 8] -> [25, 64] -> 25 + 64 = 89`
- `89 -> [8, 9] -> [64, 81] -> 64 + 81 = 145`
- `145 -> [1, 4, 5] -> [1, 16, 25] -> 1 + 16 + 25 = 42`
- `42 -> [4, 2] -> [16, 4] -> 16 + 4 = 20`
- `20 -> [2, 0] -> [4, 0 -> 4 + 0 = 4`

The number 4 has already occurred so this is a cycle. Numbers that result in a cycle via this algorithm are called *unhappy numbers*. Applying this algorithm to other positive integers eventually reaches the value of 1. This could be a considered a cycle (`1 -> 1 -> 1 ->1)` but these numbers are called *happy numbers*. For example:

- `367 -> [3, 6, 7] -> [9, 36, 49] -> 9 + 36 + 49 = 94`
- `94 -> [9, 4] -> [81, 16] -> 81 + 16 = 97`
- `97 -> [9, 7] -> [81, 49] -> 81 + 49 = 130`
- `130 -> [1, 3, 0] -> [1, 9, 0] -> 1 + 9 + 0 = 10`
- `10 -> [1, 0] -> [1, 0] -> 1 + 0 = 1`

Write a program to print out the 10^{th}, 100^{th}, 500^{th}, 1000^{th}, and 2000^{th} happy numbers one per line (A total of 5 values.) For reference the first eight happy numbers are: 1, 7, 10, 13, 19, 23, 28, 31

Input
There is no input file for this problem.

Output
Output the 10^{th}, 100^{th}, 500^{th}, 1000^{th}, and 2000^{th} happy numbers one per line.

Example Input File
NONE

Example Output To Screen
Example output is not given for this problem. Produce output described in the problem. (There will be a total of 5 lines, 1 integer per line.)

5. Hint, Hint!! (AKA, Not Quite a Quine)

Program Name: Hint.java Input File: NONE

The syntax for computer programs is weird isn't? Might as well be Greek in some cases. And reading from files?!? Don't get me started.

Oh well, here is a relatively simple problem. (No really!!) Write a program that reproduces the output shown below EXCEPT that all `@` characters are replaced by space characters, all `Q` characters are replaced by `{`'s and all `K` characters are replaced by `}`'s.

Input
There is no input file for this problem.

Output
As shown in the **Sample Text** section EXCEPT each `@` character is replaced by a single space character, each `Q` character is replaced by a single `{` character, and each `K` character is replaced by a single `}` character.

Example Input File
NONE

Sample Text (output to standard output but with replacements as described above)
```
import@java.io.*;
import@java.util.*;
public@class@template@Q
@@@@public@static@void@main(String[]@args)@throws@IOExceptionQ
@@@@@@@@Scanner@s@=@new@Scanner(new@File("hint.dat"));
@@@@@@@@int@numSets@=@s.nextInt();
@@@@@@@@s.nextLine();
@@@@@@@@for(int@i@=@1;@i@<=@numSets;@i++)Q
@@@@@@@@@@@@//do@stuff@for@data@set
@@@@@@@@K
@@@@@@@@s.close();
@@@@K
K
```

6. Mariposa Mexico or Bust

Program Name: Population.java Input File: population.dat

A biologist is monitoring the butterfly population in a field outside of Norman, Oklahoma. Each day the biologist counts the number of butterflies in the field and from the second day on, records the change in population from the previous day. The biologist wants to know the largest increase in population that occurs on consecutive days in the overall time period. The consecutive readings that have the largest sum (increase in population) can vary between the entire time period and one day.

Each integer represents the change in population for a given day. The data appears in chronological order. In other words, the first integer is the change in population that occurred between the first and second day, the second integer is the change in population that occurred between the second and third day and so forth.

For example, if the biologist records the changes over 7 days and has the following data:

```
-4 -10 6 -3 -2 0
```

The largest sum of consecutive readings in the overall time period is 6 and occurs on a single day, the 3rd day which had an increase of 6. Consider another example with data from 7 days:

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

The period of consecutive readings with the largest sum starts on day 2 and ends on day 6. The sum of the changes is `4 + -2 + 4 + 2 + 1 = 9`

If the population decreased on every day, the smallest single decrease is considered the largest total increase.

Write a program the prints out the largest sum (increase in population) of consecutive readings in the overall time period.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will consist of 2 lines:
  - The first line of a data set will be an integer M that indicates how many readings are in the data set. M will be > 0
  - The second line of data set will be M integers with a single space in between integers. Each integer represents the change in butterfly population for that day compared to the previous day. Each integer will be greater than -20000 and less than 20000

Output

- For each data set print out the largest sum of consecutive readings in the overall time period.

Example Input File

```
7
6
-4 -10 6 -3 -2 0
7
-3 4 -2 4 2 1 -3
1
12
1
-10
5
100 200 0 100 350
7
-50 -40 -100 -60 -65 -40 -10000
10
0 0 5 1 0 6 0 12 0 3
```

Example Output To Screen

```
6
9
12
-10
750
-40
27
```

7. How We Miss You, Colt McCoy

Program Name: QBRating.java Input File: qbrating.dat

Let's rate some quarterbacks! Quarterback is the premiere position in the game of American football. A quarterback's passer rating is a value that indicates how effective a quarterback has performed in a given game or season. The rating uses the following statistics:

- The number of plays the quarterback threw the ball, referred to as *ATT*.
- The total number of yards gained on passing plays the quarterback threw the ball, referred to as *YDS*.
- The number of plays the quarterback threw the ball which resulted in a touchdown, referred to as *TD*.
- The number of plays the quarterback threw the ball and the result was a completion, referred to as *COMP*.
- The number of plays the quarterback threw the ball which resulted in an interception, referred to as *INT*.

Given these statistics the formula for the NCAA passer rating is:

$$PasserRating = \frac{(8.4 \times YDS) + (330 \times TD) + (100 \times COMP) - (200 \times INT)}{ATT}$$

Write a program to compute and the passer rating for various quarterbacks.

Input

- The first line will be a single integer N that indicates the number of data sets.
- The first line in each data set will be the name of quarterback.
- The next line in each data set will be an integer ATT that indicates the number of attempts the quarterback made. ATT >= 0
- The next ATT lines will be the results of each passing attempt, one per line.
  - `INC` indicates the passing attempt resulted in an incomplete pass.
  - `INT` indicates the passing attempt resulted in an interception.
  - If a pass resulted in a completion that was not a touchdown, the line will be a single integer that indicates the number of yards gained on the play. The integer will be between -99 and 99 inclusive. (A negative value indicates the pass was completed but resulted in a loss of yardage.)
  - If a pass resulted in a completion that was a touchdown, the line will be an integer that indicates the number of yards gained on the play followed by a single space followed by `TD`. The integer will be between 1 and 99 inclusive.

Output

- For each data set output the quarterbacks name followed by a single space followed by the passer rating.
  - If ATT equals 0 print `NO RATING`
  - Otherwise print the passer rating rounded to the nearest integer.

Example Input File

```
4
Garrett Gilbert
7
13
4
INC
INC
INC
INC
7
Colt McCoy
18
21
INC
17
INC
3
22
INC
INC
4
INT
2
7
24
INC
6 TD
8
20 TD
7
Case McCoy
0
Vince Young
9
-9
INT
INT
INC
INT
45
7 TD
INT
INC
```

Example Output To Screen

```
Garrett Gilbert 72
Colt McCoy 158
Case McCoy NO RATING
Vince Young 21
```

8. One in a Million

Program Name: Rare.java Input File: rare.dat

Write a program to determine how many rare characters appear in a line of text. A rare character is a printable ASCII character that that appears in the text, but occurs less than 4% of the time compared to the total number of characters in the text.

For example given the following piece of text:

```
%^%^%^%%**&&*&^*!*&&&^***&&*&&^^%%&*!*&&:**&&&*^^^&*^^%
```

the rare characters are the exclamation point, `!`, which occurs 3.63% of the time and the colon, `:`, which occurs 1.82% of the time.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will consist of two lines.
- The first line in the data set will be an integer that indicates which data set this is.
- The second line will contain the text of the data set. Each data set only contains printable ASCII characters, those with codes between 33 and 126 inclusive.
- Each line will end with a newline character that does not count as one of the characters of the data set.

Output

- For each data set output a single line.
- Preface each line of output with `data set N, M:` where `N` is the number of the data set and `M` is the number of distinct rare characters in the data set.
- After the preface, print the distinct rare characters from the data set in ascending order based on their ASCII codes.

Example Input File **(Note, the example input has been lined wrapped. Each data set in the actual input file is on a single line.)**

```
3
1
%^%^%^%%**&&*&^*!*&&&^***&&*&&^^%%&*!*&&:**&&&*^^^&*^^%
2
98127381273812738921739871239817267287283779187282739191987238723
98172389721989273287187287373738228291827381 3g129831212391283981
2238123gggg193912390218390892833912830129380120109182398912398j1
23819239213912398129381298398081290 3j128390128390j12398091283090
3
AAAAAABAAAAAAAAAAAAAAAA
```

Example Output To Screen

```
data set 1, 2:!:
data set 2, 3:6gj
data set 3, 0:
```

9. To Ride or Not to Ride

Program Name: Ride.java Input File: ride.dat

Yet another sports question? I 'm a sensing a minor theme.

Write a program to determine if a cyclist will ride outside or go to the gym based on various conditions.

The factors used to determine if the cyclist will ride outside or go to the gym are the temperature, the time of day, whether it is raining or not, and whether the cyclist is riding alone or with a group of other cyclists.

- Temperature will be an integer between -100 and 130 inclusive.
- Time of day will be an integer between 0 and 23 inclusive. (Based on a 24-hour clock.)
- The status of rain will be expressed as `RAIN` or `NORAIN`.
- The status of a group ride will be expressed as `GROUP` or `ALONE`.

If the cyclist is riding alone, he will go outside for a ride if
- the time of day is between 6 and 18 inclusive
- AND the temperature must is between 30 and 110 inclusive unless it is raining in which case the temperature must be between 70 and 115 inclusive..

If a group is available, the cyclist will go outside for a ride if
- the time of day is between 7 and 21 inclusive
- AND the temperature must is between 20 and 105 inclusive unless it is raining in which case the temperature must be between 70 and 110 inclusive.

If the conditions above are not met, the cyclist will go the gym.

Input
- The first line will be a single integer N that indicates the number of data sets.
- Each data set will be a single line of the form `TMP TM RN GRP`
  - `TMP` will be an integer between -100 and 130 inclusive, `TM` will be an integer between 0 and 23 inclusive, `RN` will be either `RAIN` or `NORAIN`, `GRP` will be either `GROUP` or `ALONE`.

Output
- For each data set output a single line.
- Each line of output is prefaced with `data set N:` where `N` is the number of the data set.
- After the preface print `RIDE` if the cyclist will go for a ride outside, otherwise print `GYM`.

Example Input File
```
3
110 10 NORAIN ALONE
50 16 RAIN ALONE
70 19 RAIN GROUP
```

Example Output To Screen
```
data set 1: RIDE
data set 2: GYM
data set 3: RIDE
```

10. Casey at the Bat

Program Name: Slugging.java **Input File: slugging.dat**

Slugging percentage is a statistic used in baseball to rate how powerful a hitter is. Slugging percentage is calculated by dividing the total bases a hitter has obtained by the number of at bats not including the number of times the hitter walked.

Total bases is equal to the number of singles the player hit plus the number of doubles the player hit times two plus the number of triples the player hit times three plus the number of homeruns the player hit times four. In formula form slugging percentage is equal to

$$SLG = \frac{(1B) + (2 \times 2B) + (3 \times 3B) + (4 \times HR)}{AB}$$

where *SLG* is slugging percentage, *1B* is the number of singles, *2B* is the number of doubles, *3B* is the number of triples, and *HR* is the number of homeruns a player hits. *AB* represents the number of at bats a player had, which is the number of outs plus the number of singles, doubles, triples, and homeruns. Walks do not count towards the number of at bats.

Write a program that given the results of a player's data, prints out their slugging percentage to three decimals, rounded to the nearest thousandth.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will be a single line of characters. Each data set will have at least one character.
- The possible characters in a data set are O, S, D, T, H, W
  - O represents an out
  - S represents a single
  - D represents a double
  - T represents a triple
  - H represents a home run
  - W represents a walk

Output

- For each data set print out one line with the slugging percentage for the data set rounded to the nearest thousandths.
- Output shall include the digit in the ones place and three decimals. Include trailing zeros so that there are three decimal places in the output. Thus the output will value will vary from 4.000 to 0.000 inclusive.
- If the data set has no at bats, due to the player always walking, then print out `NONE`.

Example Input File

```
5
WWWWWW
HHHHHH
OOOOOSOOOSOOWWOOOSOOOODOOOOSOOOWOWOWWOOOHOOOOSOOOWOO
SODOOOSOOWOOSOOSOODOOTOOOOHOOOOOTHSHDTHWOOOSSDDTTHHW
OOOOOOWOOOOWOOOWOOWOWOOO
```

Example Output To Screen

```
NONE
4.000
0.244
1.143
0.000
```

11. No Vuvuzelas Required

Program Name: Soccer.java **Input File: soccer.dat**

Write a program that determines the first place team from a group in the World Cup Soccer tournament. The final tournament of the Soccer World Cup competition consists of group play in a round robin tournament followed by a single elimination tournament. Groups in the initial round robin tournament are composed of four teams.

In the group round robin tournament, each team in a group plays every other team in the group one time for a total of six matches. Matches can result in a win, a loss, or a draw based on the number of goals scored. If a team scores more goals in a match, it wins the match and the other team loses the match. If each team scores the same number of goals in a match, the result is a draw. A win is worth 3 points for the winning team, a draw is worth 1 point for each team, and a loss is worth 0 points for the losing team.

The first place team is the one that has the most points in the group after the round robin tournament. If two or more teams are tied for the most points, a series of tiebreakers is used.

1. If two or more teams are tied for most points, the first tiebreaker is largest goal difference. (A team's goal difference is the sum of the goals the team scored in the group matches minus the sum of the goals the team allowed in the group matches.) The team with the largest positive goal difference among the teams tied on points is the first place.
2. If two or more teams are tied for most points and largest goal difference, the second tiebreaker is the total number of goals a team scored in the group matches. The team with the most goals scored from the teams tied with points and tied for goal difference is the first place team.
3. If there are exactly two teams tied for most points, largest goal difference, and most goals scored, the third tiebreaker goes to the team that won the head to head match between the two teams that are tied.
4. If there are three or four teams tied for most points, largest goal difference, and most goals scored, OR if there are exactly two teams tied for most points, largest goal difference, and most goals scored and there head to head match was a draw THEN first place is unresolved.

Write a program that given the results of the 6 matches in a round robin group tournament prints out the first place team in the group.

Input

- The first line will be a single integer N that indicates the number of data sets.
- Each data set will consist of six lines with the format:
  `TEAMNAME1 GOALS1 TEAMNAME2 GOALS2`
- Each line in a data set is the result of the match between `TEAMNAME1` and `TEAMNAME2`. `GOALS1` is the number of goals scored by `TEAMNAME1` in the match and `GOALS2` is the number of goals scored by `TEAMNAME2` in the match.
- Team names will consist of uppercase letters only.
- `GOALS1` and `GOALS2` will be positive integers.
- A data set will contain four distinct team names and each team will play each other team once.

Output

- For each data set output one line.
- Each line of output will be `data set [N]: [WINNINGTEAM]` where `[N]` is the number of the data set and `[WINNINGTEAM]` is the first place team from the group based on the criteria explained above.
- If there is a tie that cannot be broken, print `UNRESOLVED` in place of `[WINNINGTEAM]`.

Example Input File

```
5
USA 1 ENGLAND 1
ENGLAND 1 SOUTHAFRICA 1
USA 1 IRAN 1
SOUTHAFRICA 1 IRAN 1
ENGLAND 1 IRAN 1
SOUTHAFRICA 1 USA 1
BRAZIL 3 SPAIN 1
IRAQ 5 FRANCE 2
FRANCE 1 BRAZIL 3
BRAZIL 3 IRAQ 0
SPAIN 2 IRAQ 2
SPAIN 4 FRANCE 4
SOUTHAFRICA 5 MEXICO 4
URAGUAY 3 FRANCE 0
SOUTHAFRICA 2 URAGUAY 2
FRANCE 0 MEXICO 2
MEXICO 0 URAGUAY 1
FRANCE 2 SOUTHAFRICA 4
ARGENTINA 1 NIGERIA 0
ENGLAND 1 SWEDEN 1
SWEDEN 2 NIGERIA 1
ARGENTINA 0 ENGLAND 1
SWEDEN 1 ARGENTINA 1
NIGERIA 0 ENGLAND 0
ITALY 5 SOUTHKOREA 4
ITALY 2 RUSSIA 0
SOUTHKOREA 2 RUSSIA 0
ITALY 1 IRAN 2
SOUTHKOREA 2 IRAN 1
IRAN 0 RUSSIA 0
```

Example Output To Screen

```
data set 1: UNRESOLVED
data set 2: BRAZIL
data set 3: URAGUAY
data set 4: SWEDEN
data set 5: ITALY
```

12. Lighting the Tower

Program Name: Tower.java Input File: NONE

The University of Texas Main Building, aka The Tower, dominates the campus skyline and is a symbol of the University. To celebrate various academic and athletic achievements the windows in The Tower are lit in various patterns. We couldn't get the university to light the tower in honor of the 2011 UTCS UIL Open contest, so instead write a program to display the tower as shown below.

Input
There is no input file for this problem.

Output
As shown below.

Example Input File
None.

Example Output To Screen
```
   #####
   |||||
   |||||
   |||||
   |||||
   #####
~~~~~~~~~
|-O-*-O-|
~~~**~~~~
|-*-*-O-|
~*~~*~~~~
|-O-*-O-|
~~~~*~~~~
|-O-*-O-|
~~~~*~~~~
|-O-*-O-|
~~~~*~~~~
|-O-*-O-|
~~~~*~~~~
|-O-*-O-|
~~~~*~~~~
|-O-*-O-|
~~~~*~~~~
|*******|
~~~~~~~~~
```