UW/Microsoft Programming Competition
December 14th, 2013

NOVICE DIVISION

- Do the problems in any order you like.
- All problems have a value of 60 points.
- Your program should not print extraneous output. Follow the format exactly as given in the problem.
- There is a 5-point penalty for each incorrect submission for problems that are eventually judged correct.
- Time will be used to break point ties.

Problems	Points	Notes
1. Wake Up!		
2. And the winner is…		
3. Capitalizing words game		
4. Elections		
5. Digit sum		
6. Colorful		
7. Nucleotides		
8. Rock Paper Scissors		
9. Similar		
10. DICO		
11. Climb		
12. Survey Says!		
13. Club Demographics		
14. Breaking Bad		
15. Mariposa		
16. Thief in the Night		
17. Interstellar Travel		
18. Alien Speak		
19. Overbooked		

Good luck!

1. Wake Up!

Input File: wakeup.dat

We have to wake up on weekdays and we get to sleep in on weekends. Given a day, you will determine if it is a day to wake up or a day to sleep in.

Input
The input file consists of just one line, on which there is one word. You may assume the word provided is one of the following: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or Sunday. The first letter of the day will be capitalized and the others will be lowercase.

Output
If the word is a weekday, then the output is "Wake up, it's <DAY>!" where <DAY> is the day in the file. Otherwise, the output is "Relax, it's <DAY>!"

Example Input File
```
Monday
```

Example Output to Screen
```
Wake up, it's Monday!
```

2. And the winner is…

Input File: raffle.dat

Tim is organizing a raffle event. To keep things interesting, he has decided that the winner will be the person with the second largest raffle ticker number.

Input

The first line of the input file will contain a single integer **`n`** that indicates the number of ticket numbers that follow. The next **`n`** lines each hold a single positive integer representing a raffle ticket number that was given to a person.

Output

The winning raffle ticket number (the second largest number).

Example Input File

7
5
14
8
19
20
31
5

Example Output to screen:

20

3. Capitalizing words game

Input File: capitalizewords.dat

Given a string S and a number N, make all the characters of the first N words in the given string S uppercase.

Input
The input file consists of two lines: the first contains a string of lowercase words each separated by a single space and the second contains a number.

Output
Output the original string with all the characters of the first N words in uppercase. If N is greater than the number of words in the string, simply print the whole string in uppercase.

Example Input File
```
seattle is rainy and cold
3
```

Example Output to screen:
```
SEATTLE IS RAINY and cold
```

4. Elections

Input File: elections.dat

It's time to elect a student body president! This year, there are only two candidates, Arlondra and Dahlia. Your job is to write a program to tally up the votes and announce the winner.

Input
The first line of the file will contain a single integer **n** that indicates the number of votes that follow. Each vote will be on its own line and will be either "Arlondra" or "Dahlia." You may assume that there will not be any misspellings or other words and that the first letter will always be capitalized and the other letters lowercase.

Output
Your output will consist of 3 lines: one with the vote tally for Arlondra, the next with the vote tally for Dahlia and the final one with the winner. The tally lines will start by the candidate's name followed by a colon, a space and the total number of votes received by that candidate. The final output line will consist of the winner's name followed by the text "is your new president!" You may assume that there will never be a tie.

Example Input File
```
6
Dahlia
Arlondra
Arlondra
Dahlia
Arlondra
Arlondra
```

Example Output to Screen
```
Arlondra: 4
Dahlia: 2
Arlondra is your new president!
```

5. Digit sum

Input File: digitsum.dat

Digit sums are commonly used in checksum algorithms. For example, they were used to verify the arithmetic operations of early computers and are now commonly used in cryptography.

Given a number, its digit sum is the sum of all its digits. For example, the digit sum of 604 is 10:

6 + 0 + 4 = 10

Input
An integer N between 0 and 999, 999, 999.

Output
The digit sum of N.

Example Input File
```
481
```

Example Output to Screen
```
13
```

6. Colorful

Input File: colorful.dat

The color of a rock provides insight into its composition and origin. For example, red colors imply that a rock was formed in an oxygenated environment such as a river channel. You have been handed a file describing the size and color of a variety of rocks. Your task is to calculate the number of unique colors present in the sample.

Input

The first line in the input file is a single integer **n** that indicates the number of lines that follow. The **n** following lines consist of the rock diameter followed by its color separated by some number of spaces. The diameters and colors have no spaces in them. The file is sorted by color (this means that rocks of the same color will always be grouped together).

Output

The number of unique colors of rocks.

Example Input File

```
6
2.5cm gray
4cm   gray
2.5cm greenish
1cm   greenish
1in   greenish
2m    red
```

Example Output to Screen

3

7. Nucleotides

Input File: nucleotides.dat

DNA consists of long chains of chemical compounds called *nucleotides*. There are four nucleotides present in DNA: Adenine (A), Cytosine (C), Guanine (G), and Thymine (T). Processing DNA computationally has many applications ranging from finding cures for diseases to determining genetic links between organisms. In this problem, you will find which is the most common nucleotide in a DNA sequence.

Input
The input file contains a single string composed only of the characters A, C, G, T in uppercase. You may assume the given string only contains letters A, C, G and T, has no spaces and that there will not be a tie for highest frequency.

Output
The letter that occurs the most number of times in the given string followed by a colon, a space and the number of times the letter appears.

Example Input File
```
GGCGCTAAAGTTTT
```

Example Output to Screen
T: 5

8. Rock Paper Scissors

Input File: rock.dat

It is quite likely you know this game, so not much explanation is required. However, just in case, here are the rules: ROCK beats SCISSORS, PAPER beats ROCK, and SCISSORS beat PAPER. Your job is to write a program that determines if you have won or lost.

Input
The first line will contain a single integer **n** that indicates the number of data sets that follow. Each data set will consist of two of the three items on one line, `ROCK, PAPER,` or `SCISSORS,` separated by a single space. The first item represents your throw, and the second your opponent's throw.

Output
For each data set, output the data set, followed by the words "`YOU WIN`", "`YOU LOSE`", or "`TIE`", with single space separation as shown.

Example Input File
```
4
ROCK SCISSORS
ROCK PAPER
SCISSORS ROCK
ROCK ROCK
```

Example Output to Screen
```
ROCK SCISSORS YOU WIN
ROCK PAPER YOU LOSE
SCISSORS ROCK YOU LOSE
ROCK ROCK TIE
```

9. Similar

Input File: similar.dat

Similar triangles have equal corresponding angles, and the corresponding sides are in proportion to each other. Given the three sides of one triangle and two sides of a similar triangle, find the 3rd side of the second triangle. All sides will be whole numbers (integers).

Input

The first line consists of the number of data sets in the file. Each subsequent line will have 5 integers: the 3 sides of one triangle and the 2 sides of the similar triangle.

Output

For each data set, print out the three sides of the similar triangle with the missing side.

Example Input File

```
3
3 4 5 6 8
10 12 10 5 6
2 3 4 6 9
```

Example Output to Screen

```
6 8 10
5 6 5
6 9 12
```

10. DICO

Input File: dico.dat

Many games of chance use 6-sided dice but others use 20-sided dice. A 20-sided solid with twenty congruent triangular faces is named an icosahedron.

In the game called DICO, one twenty-sided die (we'll call it a "dico") is rolled several times, and a winner or loser is determined by the following set of rules.

- If the first roll of the dico is a 7, 11, 15, or 20, the player wins.
- If the first roll of the dico is a 2, 3, 10, 13 or 19, the player loses.
- Otherwise, the dico is rolled repeatedly until either a 7, 10, 11, or 15 is rolled in which case the player loses, or until the original number (number that was rolled first) is rolled again, in which case the player wins.

This program will process a series of lines of data containing several dico rolls terminated by a 0 (the 0 is not counted as a roll). For each line of data, determine if the player wins or loses or if the game has not yet finished. If there is no result after a series of rolls, report "NO RESULT" with the number of rolls made.

Input

The first line will contain a single integer **n** that indicates the number of data sets that follow. Each data set consists of a single row of integers with zero as the final sentinel value.

Output

For each data set, output "WIN", "LOSS", or "NO RESULT", followed by one space, and then the number of rolls it took to determine the outcome, based on the rules stipulated above.

Example Input File

```
4
2 3 4 15 6 0
5 3 12 19 16 5 8 8 8 0
1 11 4 2 2 12 20 2 12 0
14 18 0
```

Example Output to Screen

```
LOSS 1
WIN 6
LOSS 2
NO RESULT 2
```

11. Climb

Input File: climb.dat

Given a number N, draw a triangle using the character 'C' that is N lines tall and N characters wide.

Input
The first line consists of the number of data sets in the file. Each data set consists of a single integer greater than 1 and less than 30.

Output
Print out each triangle from the data set, separated by one blank line.

Example Input file
```
3
2
5
8
```

Example Output to Screen
```
 C
CC

    C
   CC
  CCC
 CCCC
CCCCC

       C
      CC
     CCC
    CCCC
   CCCCC
  CCCCCC
 CCCCCCC
CCCCCCCC
```

12. Survey Says!

Input File: survey.dat

You have been hired to stuff surveys into a cluster of mailboxes numbered sequentially starting with one according to these very unusual instructions:

- Survey A goes into all mailboxes
- Survey B goes into box 2 and every other one after that, i.e., 4, 6, 8, etc.
- Survey C goes into box 3 and every third one after that, i.e., 6, 9, 12, etc.
- Continue this pattern for the remaining surveys

For example, if you have 16 mailboxes in the cluster and 5 surveys to stuff, after you are finished, box 6 will have 3 surveys in it: A, B and C. The completed distribution would be as follows:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	AB	AC	ABD	AE	**ABC**	A	ABD	AC	ABE	A	ABCD	A	AB	ACE	ABD

Input

The first line will contain a single integer **n** that indicates the number of data sets that follow.
Each data set will consist of two integers: M (# mailboxes) and S (# surveys) with S<=M.

Output

Each data set will have five output statements, labeled, formatted and spaced as shown below:

- Which box (or boxes) contains the most number of surveys
- The number of surveys in box #N
- All boxes that only contain survey A
- All boxes that contain exactly three surveys
- The total number of surveys stuffed

Example Input File

```
3
16 5
18 7
23 23
```

Example Output to Screen

```
Box that contains the most surveys: 12
Box 5 contains 2 surveys.
Boxes that contain only survey A: 1 7 11 13
Boxes that contain exactly three surveys: 4 6 8 10 15 16
Total number of surveys stuffed: 36

Box that contains the most surveys: 12
Box 7 contains 2 surveys.
Boxes that contain only survey A: 1 11 13 17
Boxes that contain exactly three surveys: 4 8 10 14 15 16
Total number of surveys stuffed: 45

Box that contains the most surveys: 12 18 20
Box 23 contains 2 surveys.
Boxes that contain only survey A: 1
Boxes that contain exactly three surveys: 4 9
Total number of surveys stuffed: 76
```

13. Club Demographics

Input File: demographics.dat

You organize a national club for people interested in walking on the moon. In your capacity as club organizer you have started to gather and analyze information about the club members so you can better understand their abilities and interests.

Input
The first line will contain a single integer **n** that indicates the number of data sets that follow. The **n** following lines consist of a person's username, the city they live in, and their age. You may assume there are no spaces in usernames or city names.

Output
The name of the city with the youngest average age followed by a colon and that city's average age.

Example Input File
```
6
foohiker   Chicago 27
ashleigh   Seattle 17
bloodlord  Seattle 16
dudeman420 Chicago 30
tealeaf    Seattle 15
pyjamas23  Denver  12
```

Example Output to Screen
```
Denver: 12.0
```

14. Breaking Bad

Input File: bad.dat

In the mega hit AMC TV show, ***Breaking Bad***, the opening credits in each episode show chemical abbreviations in each person's name. Given a file called "elements.dat", which contains all 118 chemical element abbreviations, and a data file called "bad.dat" which contains the cast names, first and last for each, output each name with the "best" single chemical element abbreviation shown in the name.

For example, if someone's name was "Breaking Bad", you would output "***B_reaking B_ad***", with an underscore following each chemical abbreviation. The "best" element to show is the one that occurs first alphabetically in each name. To clarify, the word "Breaking" contains five possible element abbreviations – "B", "Br", "I", "In", and "k". Only the first in alpha order, "B", is used. In the word "Bad", "B" is used instead of "Ba" since it is first alphabetically.

Input

The "elements.dat" file contains all 118 elements (see full listing on next page), one on each line, starting with "1 - H - Hydrogen", and ending with "118 - Uuo - Ununoctium".

The "bad.dat" file contains several names of the Breaking Bad cast members, one on each line, such as "Brian Cranston", "Aaron Paul", "Anna Gunn", and so on. The first line will contain a single integer **n** that indicates the number of names that follow.

Output

For each name, output the person's first and last name, **exactly as shown below**, with the "best" chemical abbreviation inserted in each one, according to the specifications listed above. If no abbreviation exists for a name, output the name in its original form as in the name "RJ" in the sample file.

Elements File

```
1 - H - Hydrogen
2 - He - Helium
3 - Li - Lithium
4 - Be - Beryllium
5 - B - Boron
.
.
.
114 - Fl - Flerovium
115 - Uup - Ununpentium
116 - Lv - Livermorium
117 - Uus - Ununseptium
118 - Uuo - Ununoctium
```

Example Input File

```
5
Brian Cranston
Aaron Paul
RJ Mitte
Anna Gunn
Dean Morris
```

Example Output to Screen

```
B_rian C_ranston
AAr_on P_aul
RJ MI_tte
AN_na GU_nn
DeaN_ Mo_rris
```

Elements.dat for Breaking Bad

1 - H - Hydrogen
2 - He - Helium
3 - Li - Lithium
4 - Be - Beryllium
5 - B - Boron
6 - C - Carbon
7 - N - Nitrogen
8 - O - Oxygen
9 - F - Fluorine
10 - Ne - Neon
11 - Na - Sodium
12 - Mg - Magnesium
13 - Al - Aluminum
14 - Si - Silicon
15 - P - Phosphorus
16 - S - Sulfur
17 - Cl - Chlorine
18 - Ar - Argon
19 - K - Potassium
20 - Ca - Calcium
21 - Sc - Scandium
22 - Ti - Titanium
23 - V - Vanadium
24 - Cr - Chromium
25 - Mn - Manganese
26 - Fe - Iron
27 - Co - Cobalt
28 - Ni - Nickel
29 - Cu - Copper
30 - Zn - Zinc
31 - Ga - Gallium
32 - Ge - Germanium
33 - As - Arsenic
34 - Se - Selenium
35 - Br - Bromine
36 - Kr - Krypton
37 - Rb - Rubidium
38 - Sr - Strontium
39 - Y - Yttrium
40 - Zr - Zirconium
41 - Nb - Niobium
42 - Mo - Molybdenum
43 - Tc - Technetium
44 - Ru - Ruthenium
45 - Rh - Rhodium
46 - Pd - Palladium
47 - Ag - Silver
48 - Cd - Cadmium
49 - In - Indium
50 - Sn - Tin
51 - Sb - Antimony
52 - Te - Tellurium
53 - I - Iodine
54 - Xe - Xenon
55 - Cs - Cesium
56 - Ba - Barium
57 - La - Lanthanum
58 - Ce - Cerium
59 - Pr - Praseodymium
60 - Nd - Neodymium
61 - Pm - Promethium
62 - Sm - Samarium
63 - Eu - Europium
64 - Gd - Gadolinium
65 - Tb - Terbium
66 - Dy - Dysprosium
67 - Ho - Holmium
68 - Er - Erbium
69 - Tm - Thulium
70 - Yb - Ytterbium
71 - Lu - Lutetium
72 - Hf - Hafnium
73 - Ta - Tantalum
74 - W - Tungsten
75 - Re - Rhenium
76 - Os - Osmium
77 - Ir - Iridium
78 - Pt - Platinum
79 - Au - Gold
80 - Hg - Mercury
81 - Tl - Thallium
82 - Pb - Lead
83 - Bi - Bismuth
84 - Po - Polonium
85 - At - Astatine
86 - Rn - Radon
87 - Fr - Francium
88 - Ra - Radium
89 - Ac - Actinium
90 - Th - Thorium
91 - Pa - Protactinium
92 - U - Uranium
93 - Np - Neptunium
94 - Pu - Plutonium
95 - Am - Americium
96 - Cm - Curium
97 - Bk - Berkelium
98 - Cf - Californium
99 - Es - Einsteinium
100 - Fm - Fermium
101 - Md - Mendelevium
102 - No - Nobelium
103 - Lr - Lawrencium
104 - Rf - Rutherfordium
105 - Db - Dubnium
106 - Sg - Seaborgium
107 - Bh - Bohrium
108 - Hs - Hassium
109 - Mt - Meitnerium
110 - Ds - Darmstadtium
111 - Rg - Roentgenium
112 - Cn - Copernicium
113 - Uut - Ununtrium
114 - Fl - Flerovium
115 - Uup - Ununpentium
116 - Lv - Livermorium
117 - Uus - Ununseptium
118 - Uuo - Ununoctium

15. Mariposa

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
```

16. Thief in the Night

Input File: knapsack.dat

This is an example of the Knapsack problem, a classic in computer science!

Here's the scenario: A burglar breaks into a house carrying a knapsack that can carry a limited weight of stolen items and wants to maximize the value he steals. There are many items to choose from, but he wants to optimize his haul. For example, his capacity might be 5 pounds. Let's also say he encounters only three items: an object that weighs 3 pounds which has a value of $5, a second object that weighs 2 pounds at $3 value, and a third object of weight 1 and value $4.

A brute force method involves trying every single combination. Clearly the knapsack could take each individual item by itself with room to spare, but could not take all three…too much weight. So you try two at a time. Items 1 and 2 combine for a weight of 5, with a combined value of $8. Items 2 and 3 weigh 3 pounds total with a value of $7. Items 1 and 3 weigh only 4 pounds, but have the best value of $9, so there is your best combination.

For the contest, you can use this simple brute force approach but note that it can be slow on a large dataset. A technique called dynamic programming can be really useful to speed things up. Look it up after the contest!

Input

The first line will contain a single integer **`n`** that indicates the number of data sets that follow. Each data set will start with a single integer **`k`** denoting the total weight capacity of the knapsack, and then another integer **`i`** that shows the total number of items available. Below that are listed the value and weight of each of the **`i`** items. There will be no more than 100 items to choose from, and the knapsack capacity will be no more than 100 pounds.

Output

For each data set, output the total capacity of the knapsack, the total weight of the items taken, the total value taken, and an ordered list of the items taken, as found in the data file, with the first item being #1, the next #2, etc., in the exact form you see below, with single space separation as shown and one blank line after each complete output set.

Example Input File

```
2
5
3
5 3
3 2
4 1
6
5
1 5
6 4
4 3
7 2
3 1
```

Example Output to Screen

```
knapsack capacity = 5
total weight = 4
total value = 9
Item #1 - v=5: w=3
Item #3 - v=4: w=1

knapsack capacity = 6
total weight = 6
total value = 14
Item #3 - v=4: w=3
Item #4 - v=7: w=2
Item #5 - v=3: w=1
```

17. Interstellar Travel

Input File: interstellartravel.dat

The Spiral Arm of this galaxy is vast, and trading within it takes resources. Pilot BeBob needs to buy ekti fuel (a special hydrogen fuel harvested from gas giant planets by Roamer skyminers) for his Ildiran stardrive engine (which allows for travel faster than the speed of light) for his upcoming trading voyage. He wants to buy the least amount he needs to, in order to save funds for a new and improved stardrive engine he wants to install on his ship, which the Roamer engineers have recently developed. To do this, he needs to find a path between his starting and ending location that will take him the least distance, therefore the least amount of ekti. Luckily, there are no speed limits in this quadrant of space, so less distance always implies less fuel. BeBob needs you to tell him the **distance** along the **shortest route** he can take between two locations so he knows how much money he will need to purchase enough ekti for the trip! He also knows that **you** don't get out much, you ***pizza-under-the-door-coder-you***, so he has given you a set of paths from which you can choose to help him get to his destination efficiently. For the purposes of this exercise, you may assume that a path will always exist between the ship's location and its destination.

Input

- The first line will contain a single integer **n** that indicates the number of data sets to follow.
- Each data set will consist of:
  - A list of locations that exist on BeBop's map, delimited by spaces (maximum 10)
  - An integer b representing the number paths on BeBob's map
  - A list of b paths delimited by new lines. Each line will have:
    - A string representing the name of a first location
    - A string representing the name of a second location
    - An integer representing the distance between these two locations
  - A string representing the ship's current location
  - A string representing the ship's destination

Output

For each data set in the input, output a single integer representing the shortest distance between the ship's current location and its destination. Output format should be exactly as shown.

Example Input File

```
3
M-87 XN385 Jupiter SiriusB
5
M-87 XN385 4
Jupiter M-87 5
Jupiter SiriusB 700
M-87 SiriusB 50
XN385 SiriusB 8
M-87 SiriusB
CenA CenB CX8 D-Galaxy Nebulae
6
CenA CenB 1
CenA CX8 7
CenA Nebulae 3
CenB Nebulae 6
CX8 Nebulae 2
Nebulae D-Galaxy 3
CenA D-Galaxy
Earth Theroc Ildira Llaro Rheindic
6
Earth Theroc 1
Earth Ildira 2
Earth Rheindic 3
Rheindic Llaro 4
Ildira Rheindic 1
Theroc Ildira 1
Earth Llaro
```

Example Output to Screen

```
M-87 to SiriusB:12
CenA to D-Galaxy:6
Earth to Llaro:7
```

18. Alien Speak

Input File: alienspeak.dat

Quandrea has discovered how to count in a different language, an alien language, which she has detected through her radio astronomy research. She has discovered that the aliens count by saying a phrase, and the length of that phrase is the count. However, due to the aliens being a fair distance away, Quandrea is picking up a lot of random noise and other signals in her alien tap. Luckily, the aliens send these messages twice for verification. Since the noise is random, the common counting phase between the two signals will be the **longest possible common string** between the two alien signals.

Here is an example of Quandrea's findings. Within the two signals she receives, there is embedded random noise. So if she received…

xvfzzzzqq**tan**arnae**fb** **x**shkin**vfztanf**thetrout**b**

…the underlined portions make up a phrase common to both once the noise is removed. Since this is alien counting, the length of this phrase is 9, so the number counted is 9!

Quandrea needs you to make a translator that will take these two signals and return the number that the aliens counted.

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - Two strings of speech from Quandrea's tap delimited by spaces. The translator has to figure out what is noise in each string and remove it. There can be more than one extraneous noise signal per word but you should count as little of the strings as noise as possible. Signals will only be in the form of lowercase letters [a-z].

Output

For each data set in the input, output a single integer, each on a separate line as shown below, representing the count hidden in the alien message.

Example Input File

```
4
lagqandr lxxqan
arbbeez axrbez
applesnnmbbareyubbnm applezzdeqqsareyum
hairrfgfghryjinerm rqzzzxvjkhaloiryqzzjimnneerr
```

Example Output to Screen

```
4
5
12
10
```

19. Overbooked

Input File: overbooked.dat

Being the ultimate socialite, Miss Miranda is very busy and has committed to attend too many events. Her schedule is overbooked and she needs you, her personal assistant, to help her schedule the maximum number of events she can. Events cannot overlap, and with consecutive events the first must end strictly **before** the second begins. Clearly she cannot possibly be in two places at the same time, however she can move quite quickly, even if one event starts just a minute after another finishes!

Input

- The first line will contain a single integer `n` that indicates the number of data sets to follow.
- Each data set will consist of:
  - A line containing an integer `b` that indicates the number of events Miss Miranda has scheduled
  - A line containing a space-separated list of events in the format "(start_time, end_time)". Where start_time and end_time will be in the form HH:MM on a 24 hour clock. You may assume that Miss Miranda will have no events scheduled starting before 00:00 or ending after 23:59. Start times are guaranteed to occur before end times within the same event.

Output

For each data set in the input, output a single line of events, in the format (start_time, end_time), in the order in which Miss Miranda can attend them, with single space separation as shown.

Example Input File

```
3
5
(10:00, 13:50) (10:45, 11:00) (10:00, 10:30) (11:01, 13:50) (00:00, 10:00)
3
(01:00, 02:50) (02:45, 03:00) (17:00, 22:30)
4
(01:00, 02:50) (02:45, 03:00) (17:00, 22:29) (22:30, 23:30)
```

Example Output to Screen

```
(00:00, 10:00) (10:45, 11:00) (11:01, 13:50)
(01:00, 02:50) (17:00, 22:30)
(01:00, 02:50) (17:00, 22:29) (22:30, 23:30)
```