Problem A. All about that base

Source file name:	allabout.c, allabout.cpp, allabout.java
Input:	`Standard`
Output:	`Standard`

The *base* (or *radix*) of a positional numeral system is the number of symbols that can be used to represent a number in that system. The base 10 system (also known as decimal) uses 10 distinct symbols: 0, 1, ... , 9. For example, we interpret the number 72345 as:

$$7 \times 10^4 + 2 \times 10^3 + 3 \times 10^2 + 4 \times 10^1 + 5 \times 10^0$$

Photo by Ronald Woan

This example illustrates that in base 10 the symbol at place $P \geq 0$ (starting from the right) is multiplied by 10^P to get its value. More generally, in base B we use B symbols to represent 0, . . . , $B-1$, and the symbol at the P^{th} place is multiplied by B^P to get its value.

Other bases commonly used in computation include base 2 (or binary, using symbols 0 and 1), base 8 (or octal, using symbols $0-7$), and base 16 (or hexadecimal, using symbols $0-9$ and $a-f$). In bases higher than 10, letters represent the higher values. Thus in hexadecimal $a-f$ represent the decimal values $10-15$, and in bases ≥ 36 the letter z represents the decimal value 35.

Your job is to determine the bases in which given arithmetic expressions are valid. We define an expression as *valid* in base B if two conditions are true. First, all the operands used are interpretable in base B as having values in the decimal range $[1, 2^{32}-1]$. Second, the expression is true. Any arbitrary expression might be valid in zero, one, or more bases. In this problem we will only consider bases $1-36$, where base 1 is unary.

Note that following the convention listed above, unary would consist of a single symbol: 0. In this problem, unary numbers use the symbol 1 rather than 0 (think "tally marks"). E.g., 111 in unary is equivalent to the decimal number 3 and 1111111 in unary is equivalent to the decimal number 7.

Input

Input for this problem starts with a line containing an integer $0 \leq N \leq 20$. The following N lines each contain an arithmetic expression with the following form:

$$X \ op \ Y \ = Z$$

where X, Y , and Z are positive, whole numbers consisting of 1 to 100 symbols from the set $0-9$ and $a-z$, and *op* is one of the four operators +, -, *, /. For each statement there is at least one base $1 \leq B \leq 36$ such that X, Y , and Z can all be interpreted in base B as having values in the decimal range $[1, 2^{32}-1]$.

Output

For each expression, list the bases in which the expression is valid (sorted in ascending base order) or the word "invalid" if the expression not valid in any of the bases $1-36$. Use symbols $1-9$, then $a-z$, then 0 to represent bases $1-36$ (with the last symbol, 0, representing base 36).

Example

Input	Output
8 6ef + d1 = 7c0 3 / 2 = 1 444 / 2 = 222 10111 * 11 = 1000101 10111 * 11 = 111221 5k - 1z = 46 1111111111 - 1111111 = 111 2048 - 512 = 1536	g invalid 56789abcdefghijklmnopqrstuvwxyz0 2 3456789abcdefghijklmnopqrstuvwxyz0 invalid 1 a

Problem B. Bobby's Bet

Source file name:	bobby.c, bobby.cpp, bobby.java
Input:	`Standard`
Output:	`Standard`

Bobby and Betty have a bet. Betty bets Bobby that he cannot roll an S-sided die (having values 1 through S) and obtain a value $\geq R$ on at least X out of Y rolls. Betty has a variety of dice with different numbers of sides S, and all her dice are fair (for a given die, each side's outcome is equally likely). In order to observe statistically rare events while still giving Bobby a reason to bet, Betty offers to pay Bobby W times his bet on each encounter. For example, suppose Betty bets Bobby 1 bitcoin that he can't roll at least a 5 on a 6-sided die at least two out of three times; if Bobby does, she would give him $W = 3$ times his initial bet (i.e. she would give him 3 bitcoins). Should Bobby take the bet (is his expected return greater than his original bet)?

Photo by StarsApart

Input

Input begins with an integer $1 \leq N \leq 10000$, representing the number of cases that follow. The next N lines each contain five integers, R, S, X, Y , and W. Their limits are $1 \leq R \leq S \leq 20$, $1 \leq X \leq Y \leq 10$, and $1 \leq W \leq 100$.

Output

For each case, output "yes" if Bobby's expected return is greater than his bet, or "no" otherwise. Bobby is somewhat risk averse and does not bet if his expected return is equal to his bet.

Example

Input	Output
2 5 6 2 3 3 5 6 2 3 4	no yes
3 2 2 9 10 100 1 2 10 10 1 1 2 10 10 2	yes no yes

Problem C. Cryptographer's Conundrum

Source file name:	conundrum.c, conundrum.cpp, conundrum.java
Input:	`Standard`
Output:	`Standard`

The walls of the corridors at the Theoretical Computer Science group (TCS) at KTH are all but covered with whiteboards. Some of the faculty members are cryptographers, and like to write cryptographic puzzles on the whiteboards. A new puzzle is added whenever someone discovers a solution to the previous one.

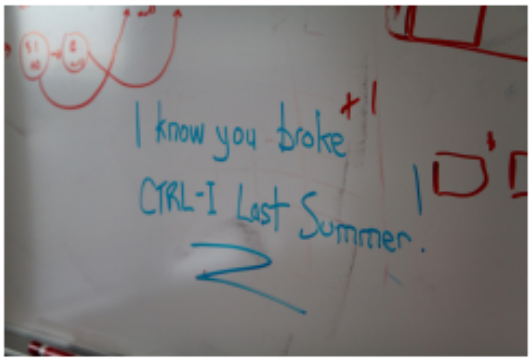


Photo by Alan Wu

When Per walked in the corridor two weeks ago, he saw that the newest puzzle read "`GuvfVfNGrfg`". After arriving at his computer, he quickly figured out that this was a simple ROT13 encryption of "`ThisIsATest`".

The series of lousy puzzles continued next week, when a new puzzle read "`VmkgdGFyIHPDpGtlcmhldGVuIHDDpSBzdMO2cnN0YSBhbGx2YXIK`". This was just base64-encoded text! "Enough with these pranks", Per thought; "I'm going to show you!"

Now Per has come up with a secret plan: every day he will erase one letter of the cipher text and replace it with a different letter, so that, in the end, the whole text reads "`PerPerPerPerPerPerPer`". Since Per will change one letter each day, he hopes that people will not notice.

Per would like to know how many days it will take to transform a given cipher text into a text only containing his name, assuming he substitutes one letter each day. You may assume that the length of the original cipher text is a multiple of 3.

For simplicity, you can ignore the case of the letters, and instead assume that all letters are upper-case.

Input

The first and only line of input contains the cipher text on the whiteboard. It consists of at most 300 upper-case characters, and its length is a multiple of 3.

Output

Output the number of days needed to change the cipher text to a string containing only Per's name.

Example

Input	Output
SECRET	4

Problem D. Road Work

Source file name:	road.c, road.cpp, road.java
Input:	`standard`
Output:	`standard`

Per is repairing roads. The job is concentrated on roads with one lane in each direction. Thus, when Per closes down the lane in one direction, all traffic has to go through the other lane. This is done by allowing only one direction of travel at any time. Per is often assigned the task of directing the traffic through this lane.

No car drives before being given a "go" signal from Per, and all the cars drive through the maintained segment at the same speed. Because there is only one lane, cars in one direction must leave the segment before cars in the other direction can enter. For safety reasons, cars driving in the same direction have to keep a distance of at least 3 seconds between each other.

For example, if cars `A` and `B` arrive at the west endpoint at second 10, Per can let them go at earliest second 10 and 13 in the order they arrived. If it, in this example, takes 8 seconds to pass and car `C` arrives at the east endpoint at second 17, then car `C` has to wait 4 seconds until Per lets it go at second 21.

There is a problem of drivers getting irritated with Per; they think they have to stop for too long. Per has been logging how long they can bear to wait before they get irritated. One day, to be able to evaluate his work, Per noted down when the cars arrived at the two endpoints of the segment. Per's question is the following: what is the least number of drivers that can be irritated? We assume that a driver gets irritated if the time between the moment he arrives at the maintained segment and the moment he is actually given the "go" exceeds his irritation time limit.

Input

The first line of the input contains two integers t and n ($4 \leq t \leq 180$ and $1 \leq n \leq 250$), where t is the time in seconds needed for a car to pass the segment under maintenance, and n is the total number of cars arriving at the segment. The following n lines describe the cars. The i-th line contains the description of the i-th car in the following format:

- one character d, being `W` for cars arriving at the west endpoint of the segment, and `E` for the ones that arrive at the east endpoint; and
- two integers a and r ($0 \leq a < 86\,400$ and $0 \leq r \leq 3\,600$), where a denotes the arrival time in seconds after midnight, and r denotes the time in seconds it takes for the driver to get irritated.

The cars arrive in the order specified in the input and they cannot overtake each other. In particular, a car whose driver is already irritated has to stay in the queue until eventually receiving the "go" and passing the maintained segment.

Output

Output one line with the least possible number of irritated drivers.

Example

Input	Output
8 3 W 10 0 W 10 3 E 17 4	0
100 5 W 0 200 W 5 201 E 95 1111 E 95 1 E 95 11	1

Problem E. Cantina of Babel

Source file name:	cantina.c, cantina.cpp, cantina.java
Input:	`Standard`
Output:	`Standard`

Characters in Star Wars each speak a language, but they typically understand a lot more languages that they don't or can't speak. For example, Han Solo might speak in Galactic Basic and Chewbacca might respond in Shyriiwook; since they each understand the language spoken by the other, they can communicate just fine like this.

Photo by Brickset

We'll say two characters can *converse* if they can exchange messages in both directions. Even if they didn't understand each other's languages, two characters can still converse as long as there is a sequence of characters who could translate for them through a sequence of intermediate languages. For example, Jabba the Hutt and R2D2 might be able to converse with some help. Maybe when Jabba spoke in Huttese, Boba Fett could translate to Basic, which R2D2 understands. When R2D2 replies in Binary, maybe Luke could translate to Basic and then Bib Fortuna could translate back to Huttese for Jabba.

In Star Wars Episode IV, there's a scene with a lot of different characters in a cantina, all speaking different languages. Some pairs of characters may not be able to converse (even if others in the cantina are willing to serve as translators). This can lead to all kinds of problems, fights, questions over who shot first, etc. You're going to help by asking some of the patrons to leave. The cantina is a business, so you'd like to ask as few as possible to leave. You need to determine the size of the smallest set of characters S such that if all the characters in S leave, all pairs of remaining characters can converse.

For example, in the first sample input below, Chewbacca and Grakchawwaa can converse, but nobody else understands Shyriiwook, so they can't converse with others in the bar. If they leave, everyone else can converse. In the second sample input, Fran and Ian can converse, as can Polly and Spencer, but no other pairs of characters can converse, so either everyone but Polly and Spencer must leave or everyone but Fran and Ian.

Input

Input starts with a positive integer, $1 \leq N \leq 100$, the number of characters in the cantina. This is followed by N lines, each line describing a character. Each of these N lines starts with the character's name (which is distinct), then the language that character speaks, then a list of 0 to 20 additional languages the character understands but doesn't speak. All characters understand the language they speak. All character and language names are sequences of 1 to 15 letters (a-z and A-Z), numbers, and hyphens. Character names and languages are separated by single spaces.

Output

Print a line of output giving the size of the smallest set of characters S that should be asked to leave so that all remaining pairs of characters can converse.

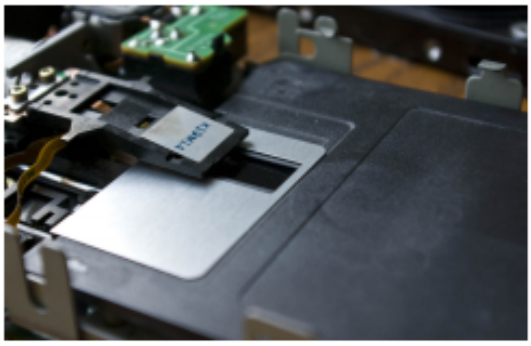
Example

Input	Output
7 Jabba-the-Hutt Huttese Bib-Fortuna Huttese Basic Boba-Fett Basic Huttese Chewbacca Shyriiwook Basic Luke Basic Jawaese Binary Grakchawwaa Shyriiwook Basic Jawaese R2D2 Binary Basic	2
6 Fran French Italian Enid English German George German Italian Ian Italian French Spanish Spencer Spanish Portugese Polly Portugese Spanish	4

Problem F. Floppy Music

Source file name:	floppy.c, floppy.cpp, floppy.java
Input:	`Standard`
Output:	`Standard`

Your friend's newest hobby is to play movie theme songs on her freshly acquired floppy drive organ. This organ is a collection of good old floppy drives, where each drive has been tampered with to produce sound of a unique frequency. The sound is produced by a step motor that moves the read/write head of the floppy drive along the radial axis of the drive's spin disk. The radial axis starts in the center of the spin disk and ends at the outer edge of the spin disk.



Antoine Taveneaux, cc-by-sa

The sound from one drive will play continuously as long as the read/write head keeps moving in one direction; when the head changes direction, there is a brief pause of 1fs—one floppysecond, or about 100 microseconds. The read/write head must change direction when it reaches either the inner or the outer end point of the radial axis, but it can also change direction at any other point along this axis, as determined by your friend. You can make the head stay still at any time and for as long as you wish. The starting position of the read-write head can be chosen freely.

Your friend is a nutcase perfectionist, and will not accept any pauses where there are not supposed to be any; nor will she accept sound when there is meant to be silence. To figure out whether a given piece of music can be played—perfectly—on her organ, she has asked for your help.

For each frequency, you are given a list of intervals, each describing when that particular frequency should play, and you must decide if all of the frequencies can be played as intended. You can assume your friend has enough drives to cover all the required frequencies.

Input

The first line contains an integer f, $1 \leq f \leq 10$, denoting the number of frequencies used. Then follow f blocks, on the format:

- A single line with two integers t_i , $1 \leq t_i \leq 10000$ and n_i , $1 \leq n_i \leq 100$; the number of floppyseconds it takes for the read/write head of frequency i to move between the end points of its radial axis, and the number of intervals for which frequency i should play.
- n_i lines, where the j-th line has two integers $t_{i,2j}$, $t_{i,2j+1}$, where $0 \leq t_{i,2j}, t_{i,2j+1} \leq 1000000$, indicating that the i-th frequency should start playing at time $t_{i,2j}$ and stop playing at time $t_{i,2j+1}$. You can assume that these numbers are in stricly ascending order, i.e. $t_{i,1} < t_{i,2} < \cdots < t_{i,2ni}$.

Output

If it is possible to play all the f frequencies as intended, output "`possible`". Otherwise output "`impossible`".

Example

Input	Output
1 6 2 0 4 6 12	possible
1 6 3 0 5 6 8 9 14	impossible

Problem G. Circuit Counting

Source file name:	countcircuits.c, countcircuits.cpp, countcircuits.java
Input:	`Standard`
Output:	`Standard`

Suppose you are given a sequence of N integer-valued vectors in the plane (x_i, y_i), $i = 1,\ldots, N$. Beginning at the origin, we can generate a path by regarding each vector as a displacement from the previous location. For instance, the vectors (1, 2), (2, 3), (-3, -5) form the path (0, 0),(1, 2),(3, 5),(0, 0). We define a path that ends at the origin as a *circuit*. The example just given is a circuit. We could form a path using any nonempty subset of the N vectors, while the result (circuit or not) doesn't depend on the ordering of the subset. How many nonempty subsets of the vectors form circuits?

For instance, consider the vectors $\{(1,2),(-1,-2),(1,1),(-2,-3),(-1,-1)\}$ From these vectors we can construct 4 possible subset circuits using

$$\{(1,2),(-1,-2)\}$$

$$\{(1,1),(-1,-1)\}$$

$$\{(1,2),(1,1),(-2,-3)\}$$

$$\{(1,2),(-1,-2),(1,1),(-1,-1)\}$$

Input

Input begins with an integer $N \leq 40$ on the first line. The next N lines each contain two integer values x and y forming the vector (x, y), where $|x|, |y| \leq 10$ and $(x, y) \neq (0, 0)$. Since the given vectors are a set, all vectors are unique.

Output

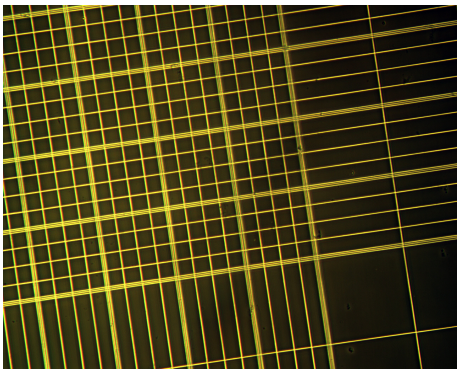
Output the number of nonempty subsets of the given vectors that produce circuits. It's guaranteed that the answer is less than 10^{10}.

Example

Input	Output
5 1 2 1 1 -1 -2 -2 -3 -1 -1	4

Problem H. How many squares?

Source file name:	howmany.c, howmany.cpp, howmany.java
Input:	`standard`
Output:	`standard`



While browsing the internet, of course using Internet Explorer without any adblocker, you have noticed a number of interesting competitions advertised in the panels on various webpages. In most of these competitions you need to answer a simple question, like how many triangles/squares/rectangles there are in a picture, or even choose the right answer out of three possibilities. Despite the simplicity of the task, it seems that there are many valuable prizes to be won. So there is definitely something to compete for!

In order to increase your chances, you decided to write a simple program that will solve the problem for you. You decided to focus first on the question "How many squares are there in the picture?", and to simplify the problem even more, you assume that the input picture consists only of a number of lines that are infinite in both directions. To be precise, we say that four lines $\ell_1, \ell_2, \ell_3, \ell_4$ in the picture form a square if lines ℓ_1 and ℓ_3 are parallel to each other and perpendicular to ℓ_2 and ℓ_4, and moreover the distance between ℓ_1 and ℓ_3 is the same as the distance between ℓ_2 and ℓ_4.

Input

The first line of the input contains a single integer n ($1 \leq n \leq 2\,000$), denoting the number of lines in the input picture. Then follow n lines, each containing a description of one line in the input picture. The line is given as a pair of distinct points lying on it. That is, the description consists of four integers x_1, y_1, x_2, y_2, each of them of absolute value at most 10 000, such that the line passes through points (x_1, y_1) and (x_2, y_2). You may assume that points (x_1, y_1) and (x_2, y_2) are different, and also that all the lines in the picture are pairwise different.

Output

Output exactly one line with one integer, denoting the total number of squares formed by the lines in the picture.

Example

Input	Output
10 0 0 1 0 0 1 1 1 0 2 2 2 0 0 0 4 1 -1 1 0 2 -2 2 2 1 1 2 2 1 1 0 2 3 1 2 2 1 3 0 2	6

Problem I. Secret Message

Source file name:	secretmessage.c, secretmessage.cpp, secretmessage.java
Input:	`Standard`
Output:	`Standard`

Jack and Jill developed a special encryption method, so they can enjoy conversations without worrrying about eavesdroppers. Here is how: let L be the length of the original message, and M be the smallest square number greater than or equal to L. Add (M - L) asterisks to the message, giving a padded message with length M. Use the padded message to fill a table of size $K \times K$, where $K^2 = M$. Fill the table in row-major order (top to bottom row, left to right column in each row). Rotate the table 90 degrees clockwise. The encrypted message comes from reading the message in row-major order from the rotated table, omitting any asterisks.

For example, given the original message 'iloveyouJack', the message length is $L = 12$. Thus the padded message is 'iloveyouJack****', with length $M = 16$. Below are the two tables before and after rotation.

i	l	o	v
e	y	o	u
J	a	c	k
*	*	*	*

*	J	e	i
*	a	y	l
*	c	o	o
*	k	u	v

Then we read the secret message as 'Jeiaylcookuv'.

Input

The first line of input is the number of original messages, $1 \leq N \leq 100$. The following N lines each have a message to encrypt. Each message contains only characters a–z (lower and upper case), and has length $1 \leq L \leq 10000$.

Output

For each original message, output the secret message.

Example

Input	Output
2 iloveyoutooJill TheContestisOver	iteiloylloooJuv OsoTvtnheiterseC

Problem J. Simon Says

Source file name:	simonsays.c, simonsays.cpp, simonsays.java
Input:	`Standard`
Output:	`Standard`

In the game "Simon Says" one person plays the role of Simon, who gives instructions to everyone else playing the game. The tricky part is that if Simon begins his instruction with "Simon says" then everyone else *must* follow the instruction (or they lose the game); if Simon gives an instruction that does not begin with "Simon says" then everyone is supposed to completely ignore the instruction (or they lose the game)!


Photo by David Amsler

Simon tries his or her best to trick the other players into following the wrong instructions. Simon might begin by saying "Simon says touch your nose." and follow this with "Stop touching your nose." Anyone who stops touching their nose loses! The last player still remaining, who has correctly followed precisely the instructions that began with "Simon says" (and only these instructions), gets to be Simon next.

As a child, you were horrible at this game. Your older siblings were always able to trick you into following the wrong instructions. Well, you will have the last laugh: now that you are a computer programmer, you can write a computer program that can help you play the game perfectly. You only need to make sure the program is able to determine which instructions to follow and which to ignore.

Are you up to the challenge? Can you craft a computer program that *never* makes any mistakes in the game? If you can, then surely fame and glory shall come your way for being the most unstoppable player of Simon Says ever!

Input

Input starts with a line containing an integer $1 \leq N \leq 1000$. Each of the next N lines is one command, of length at most 100 characters. Each command is a properly-capitalized sequence of one or more words, separated by a single space between each pair of words, ending in a period. Some commands begin with "Simon says" and others may not. If a command begins with "Simon says", there will always be another space and at least one additional word after "says". No lines contain leading or trailing space.

Output

For each line that begins with precisely "Simon says", output the rest of the line. Each line that does not begin with precisely "Simon says" should be ignored.

Example

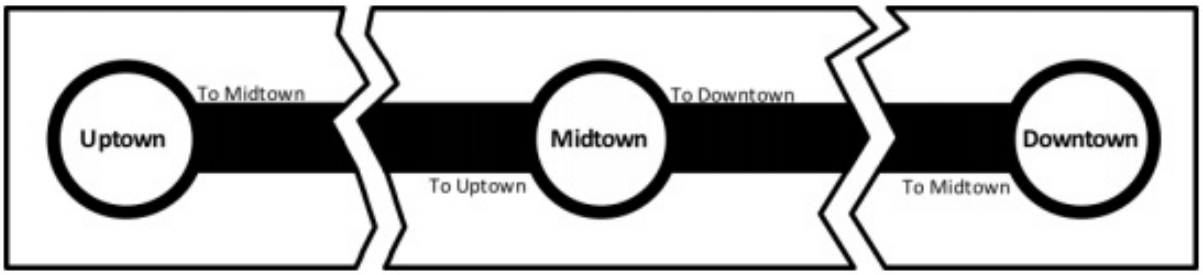
Input	Output
1 Simon says smile.	smile.
3 Simon says raise your right hand. Lower your right hand. Simon says raise your left hand.	raise your right hand. raise your left hand.
3 Raise your right hand. Lower your right hand. Simon says raise your left hand.	raise your left hand.

Problem K. Torn To Piece

Source file name:	torn2pieces.c, torn2pieces.cpp, torn2pieces.java
Input:	`Standard`
Output:	`Standard`

You have arrived in The Big City but your journey is not yet complete. You must still navigate the subway and get to your final destination. The information booth in the subway station is unattended and fresh out of maps of the subway system. On the floor you notice fragments of a map. Can you piece together enough of the map to figure out how to get to your final destination?

Each fragment of the map happens to perfectly contain a single subway station while also identifying all of the other stations that it connects to. Each connection between stations is bi-directional such that it can be travelled going either direction. Using all of the available fragments, your task is to determine the sequence of stations you must pass through in order to reach your final destination or state that there is no route if you don't have enough information to complete your journey.



Input

The first line of input has an integer, $2 \leq N \leq 32$, that identifies the number of pieces of the map that were found.

The following N lines each describe a station depicted on one of those pieces. Each of these lines starts with the name of the station they describe and is followed by a space-separated list of all of the station names that are directly connected to that station (there may be as many as $N - 1$).

The final line identifies a starting station and a destination station. The destination station is guaranteed to be different than the starting station.

Each station name is a string of up to 20 characters using only letters a–z and A–Z. It is guaranteed that there is at most one simple route (without revisiting stations) from the starting station to the destination station.

Output

Give the sequence of stations that leads from the starting station to the destination station. Separate station names with spaces. If there are not enough pieces of the map to find a route from the starting station to the destination station then output "`no route found`".

Example

Input	Output
3 Uptown Midtown Midtown Uptown Downtown Downtown Midtown Uptown Downtown	Uptown Midtown Downtown
6 A B B A D C D E D F G F E G E F A	F E D B A
4 FirstStop SecondStop SecondStop FirstStop ThirdStop FifthStop FourthStop SixthStop SixthStop FifthStop FirstStop FifthStop	no route found