Problem F: If You Can't Stand the Heat...

Boudreau never was that good of a cook. In fact he has to use a cheat sheet to make a peanut butter and jelly sandwich. In fact, the recipe looks something like:

```
bread peanut_butter jelly bread
```

Each token represents one step in the process of making the sandwich. And it is important that the steps be followed in the order listed. For example "bread jelly peanut_butter bread" is not a peanut butter and sandwich according to Boudreau's recipe. (Though he would probably still enjoy eating it.)

Most of the simple recipes that Boudreau can follow are all similar to that one. Steps that must be followed in order. However Boudreau does have the ability to follow some more complicated recipes. For instance, when he wants meat on his sandwiches then he can use the following recipe:

```
bread ( turkey | ham ) bread
```

This recipe uses the special symbol '|' which means that the chef can decide to use either the previous or next operand, but not both. So following this recipe, the cook will end up with either "bread turkey bread" or "bread ham bread". Notice also the use of parentheses, which act to group parts of the recipes, much as they do in mathematical equations. And just like in mathematical equations order of operations is important. When multiple steps are on either side of a '|' either all the steps on the left or all the steps on the right should be followed. To demonstrate this use is the following recipe:

```
bread ( peanut_butter jelly | bacon lettuce tomato ) bread
```

Following this recipe will result in "bread peanut_butter jelly bread" or "bread bacon lettuce tomato bread". So Boudreau will end up with either a PB&J or BLT.

There is one last rule in the recipes that Boudreau uses. There is a special symbol '$' which means not to do anything. In the simplest of recipes, this symbol can be ignored and therefore doesn't add anything useful. But when used with '|' it can be very useful:

```
bread $ peanut_butter $ jelly $ bread
bun patty ( cheese | $ ) bun
```

The first of the above two recipes can only be used to make the simple peanut butter and jelly sandwich, so although the '$'s don't add anything semantically to the recipe it is still a valid recipe. With the second recipe, the '$' is more important. Boudreau can either use cheese, in which case he'll make a cheeseburger ("bun patty cheese bun"), or he can choose the '$', in which case he'll make a plain hamburger ("bun patty bun").

Thibodeaux, however, is a rather more accomplished chef that Boudreau. He can cook

any of the recipes that Boudreau can cook, plus he understands recipes that use another special symbol, '*'. The symbol '*' means that the previous step can be performed any number of times. As a special case it can be performed 0 times. The '*' only applies to the immediately preceding step, which will be either a token or a sub-recipe contained in parentheses. Thibodeaux uses this symbol to make his specialty, Gumbo:

```
get_out_pot ( pick_random_item place_in_pot ) * heat_pot
```

Notice how the two steps pick_random_item and place_in_pot need to both be in parentheses in order for the '*' to be applied to both of them. Following this recipe, Thibodeaux can put an unlimited number of random items into his Gumbo.

Although Thibodeaux makes a mean Gumbo, there are still recipes that are beyond even his ability. These recipes use another rule, a 'goto'. A special step of the form <goto:*label*> is used to mean that once the recipe gets to that step, the chef should immediately continue the recipe at the step of the form <*label*>, called the target. If any goto is used, then there will be one and only one corresponding target. There may be several gotos which refer to the same target. If a target is encountered in the recipe then it is treated as a '$', in that no step should be performed.

There are many interesting properties that such recipes can have, and the following examples demonstrate a few of them:

```
bread turkey <goto:skip_pickles> pickles <skip_pickles> bread
hatch_idea <scheme> scheme <goto:scheme>
catch_roadrunner
```

There may be a step ,which is unreachable, or the recipe may fall into an infinite loop, in which case there are no valid outcomes.

Boudreau and Thibodeaux have several recipes. They have also made several attempts to follow each recipe. They want you to write a program to tell if their attempts have succeeded.

Input

The input will begin with a line containing a single integer *R* indicating the number of recipes.

The description of each recipe will be on two lines. The first line contains the name of the recipe, which may contain spaces or other punctuation. The second line contains the recipe itself. The recipe will consist of a positive number of tokens and/or the punctuation symbols '(', ')', '|', '$', and '*'. Tokens will contain only lower case letters and/or the underscore. With the exception of goto and target tokens, which will be in the format described above, with their labels containing only lower case letters and/or underscores. All tokens and punctuation will be separated by a single space.

All recipes will be validly formed. Each '*' symbol will be directly preceded by at least

one non-target, non-goto operand. All parentheses will be matched and properly nested. There will be at least one token within each pair of matching parentheses. And each goto will correspond to one and only one target.

Each recipe description will be followed by a line with a single positive integer *A* indicating the number of attempts that Boudreau and Thibodeaux made. Then *A* lines will follow each containing one attempt.

Each attempt will be a line containing a sequence of a non-negative number of steps, separated by a single space. Each step will be a string containing only lower case letters and/or underscores.

There will be no leading or trailing whitespace on any line of input.

Output

For each recipe, output a line containing:

`Boudreau and Thibodeaux attempt to make` *`recipe name`*`.`
`And for each attempt, output a line containing:`
`-->` `Attempt number` *`number`* `resulted in` *`result`*`.`

Where the valid results are "success" and "failure". Output a blank line between recipes.

Sample Input

```
5
Peanut Butter and Jelly Sandwiches
bread peanut_butter jelly bread
2
bread peanut_butter jelly bread
bread jelly peanut_butter bread
Sandwiches
bread ( peanut_butter jelly | bacon lettuce tomato ) bread
3
bread peanut_butter jelly bread
bread bacon lettuce tomato bread
bread peanut_butter jelly lettuce tomato bread
Burgers
bun patty ( cheese | $ ) bun
3
bun patty cheese bun
bun patty bun
bun patty nothing bun
Gumbo
get_out_pot ( pick_random_item place_in_pot ) * heat_pot
4
get_out_pot heat_pot
```

```
get_out_pot pick_random_item place_in_pot heat_pot
get_out_pot pick_random_item place_in_pot pick_random_item heat_pot
get_out_pot pick_random_item place_in_pot pick_random_item place_in_pot heat_pot
Turkey Sandwiches
bread turkey <goto:skip_pickles> pickles <skip_pickles> bread
2
bread turkey bread
bread turkey pickles bread
```

Sample Output

```
Boudreau and Thibodeaux attempt to make Peanut Butter and Jelly Sandwiches.
--> Attempt number 1 resulted in success.
--> Attempt number 2 resulted in failure.

Boudreau and Thibodeaux attempt to make Sandwiches.
--> Attempt number 1 resulted in success.
--> Attempt number 2 resulted in success.
--> Attempt number 3 resulted in failure.

Boudreau and Thibodeaux attempt to make Burgers.
--> Attempt number 1 resulted in success.
--> Attempt number 2 resulted in success.
--> Attempt number 3 resulted in failure.

Boudreau and Thibodeaux attempt to make Gumbo.
--> Attempt number 1 resulted in success.
--> Attempt number 2 resulted in success.
--> Attempt number 3 resulted in failure.
--> Attempt number 4 resulted in success.

Boudreau and Thibodeaux attempt to make Turkey Sandwiches.
--> Attempt number 1 resulted in success.
--> Attempt number 2 resulted in failure.
```