Doughnut Shop

Little Trevor liked coffee so much he wanted to be a cop when he grew up. He spent a lot of time pretending to direct traffic and tried to stop many people from doing bad things. He grew out of that phase, but never dropped his love of coffee and for that matter doughnuts.

Trevor likes to know which doughnut shops sell what doughnuts and what doughnuts are sold at which shops. Trevor wants to keep track of all that sort of information in his head, but due to his crippling asthma he cannot do so. Please write a program to keep track of the doughnuts for him.

He will give periodically add lists of doughnuts to a list of shops.

When Trevor gives your program a doughnut, he wants the program to return all shops with the given doughnut, and when Trevor gives your program a shop, he wants the program to return all doughnuts sold there.

Input Specification

The beginning of each operation will be an integer (either 0, 1, 2 or 3).

When an operation begins with a 1, it means Trevor is adding doughnuts to a list of shops. He will give 2 positive integer ***n***, ***m***. The value ***n*** will be the number of doughnuts and the value ***m*** will be the number of shops. All of the ***m*** shops will sell all the ***n*** different type of doughnuts. The following ***n*** lines will each contain a string (composed of at most 24 lower case letters and underscores "_") representing the names of the doughnuts to be added to the given shops **(if they don't already contain it)**. After which the following ***m*** lines will each contain a string (composed of at most 24 lower case letters and underscores "_") representing the names of the shops selling said doughnuts.

When an operation begins with a 2, it means Trevor is requesting the shops that sell a particular doughnut. The following line will contain a single string (composed of at most 24 lower case letters and underscores "_") representing the name of a doughnut.

When an operation begins with a 3, it means Trevor is requesting the doughnuts sold at a particular doughnut shop. The following line will contain a single string (composed of at most 24 lower case letters and underscores "_") representing the name of a doughnut shop.

When an operation begins with a 0, it means Trevor is done, and the program should exit successfully.

Output Specification

For each operation of type 2 output on the first line a single integer ***j*** representing the number of shops that sell the given doughnut. The following ***j*** lines should contain the name of a single shop that sells the given doughnut. The shop names should be in lexicographical order (where "_" is less than all other Latin characters).

For each operation of type 3 output on the first line a single integer ***k*** representing the number of shops that sell the given doughnut. The following ***k*** lines should contain the name of a single doughnut that is sold in the given shop. The doughnut names should be in lexicographical order (where "_" is less than all other Latin characters).

Input Output Example

Input	Output
1 3 4 chocolate_glaze chocolate chocolate_sprinkle derpy_ds dough_it_correctly doughnut_think_twice all_things_chocolate 1 2 2 jelly custard derpy_ds doughnut_do_puns 1 2 1 jelly mango dough_it_correctly 2 jelly 3 derpy_ds 3 dough_it_correctly 0	3 derpy_ds dough_it_correctly doughnut_do_puns 5 chocolate chocolate_glaze chocolate_sprinkle custard jelly 5 chocolate chocolate_glaze chocolate_sprinkle custard mango

Input	Output
1 6 2 pain cruller beignet lemon_poppy brioche bunt french_food_r_us hole_foods 1 3 4 dahi_vada vada_pav medu_vada vada_you_want darth_vada hole_foods eat_it 3 eat_it 1 2 1 lemon_poppy bunt eat_it 3 eat_it 3 does_not_exist 0	3 dahi_vada medu_vada vada_pav 5 bunt dahi_vada lemon_poppy medu_vada vada_pav 0

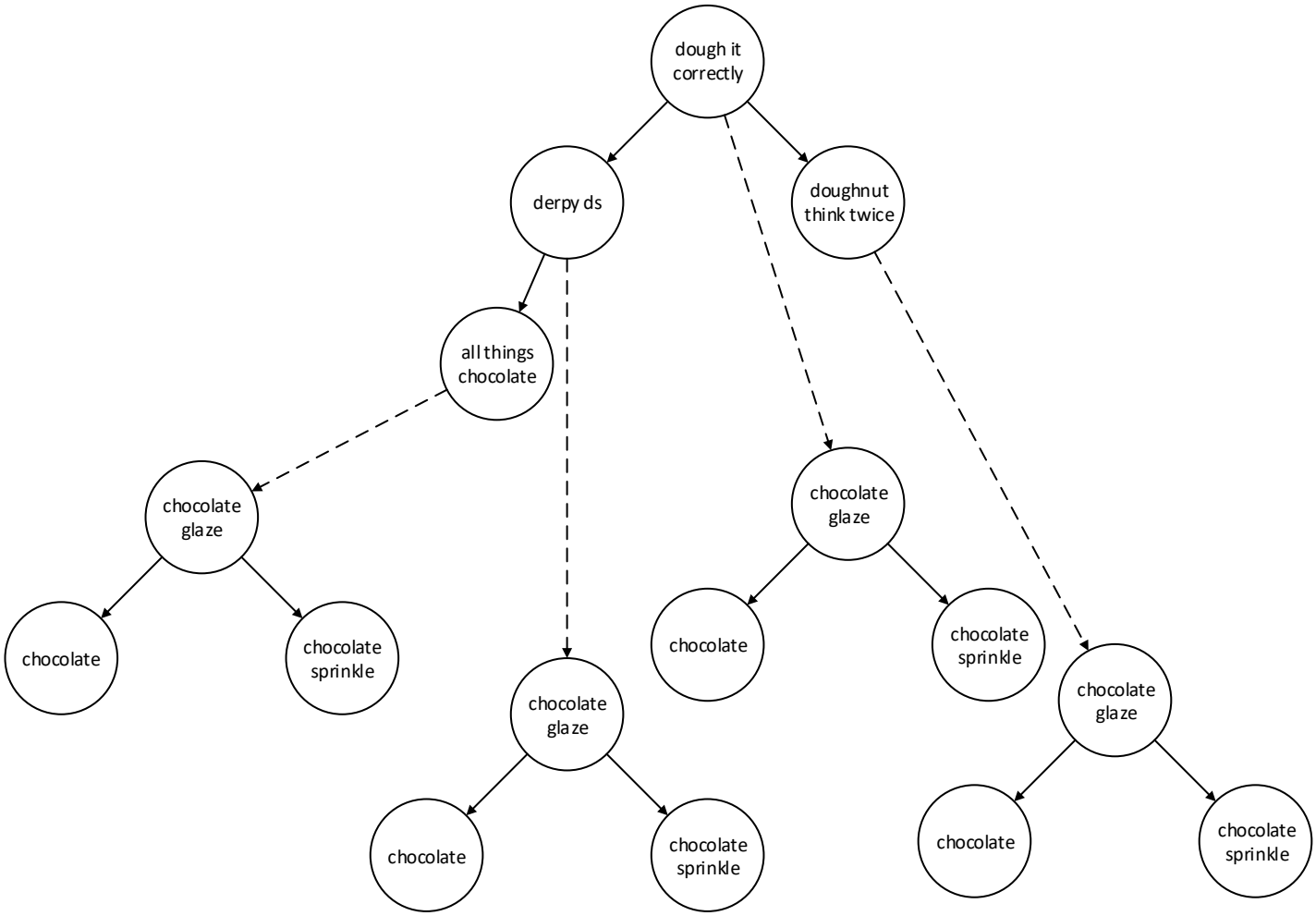
Explanation

Case 1

In the beginning there were no shops and no doughnuts…

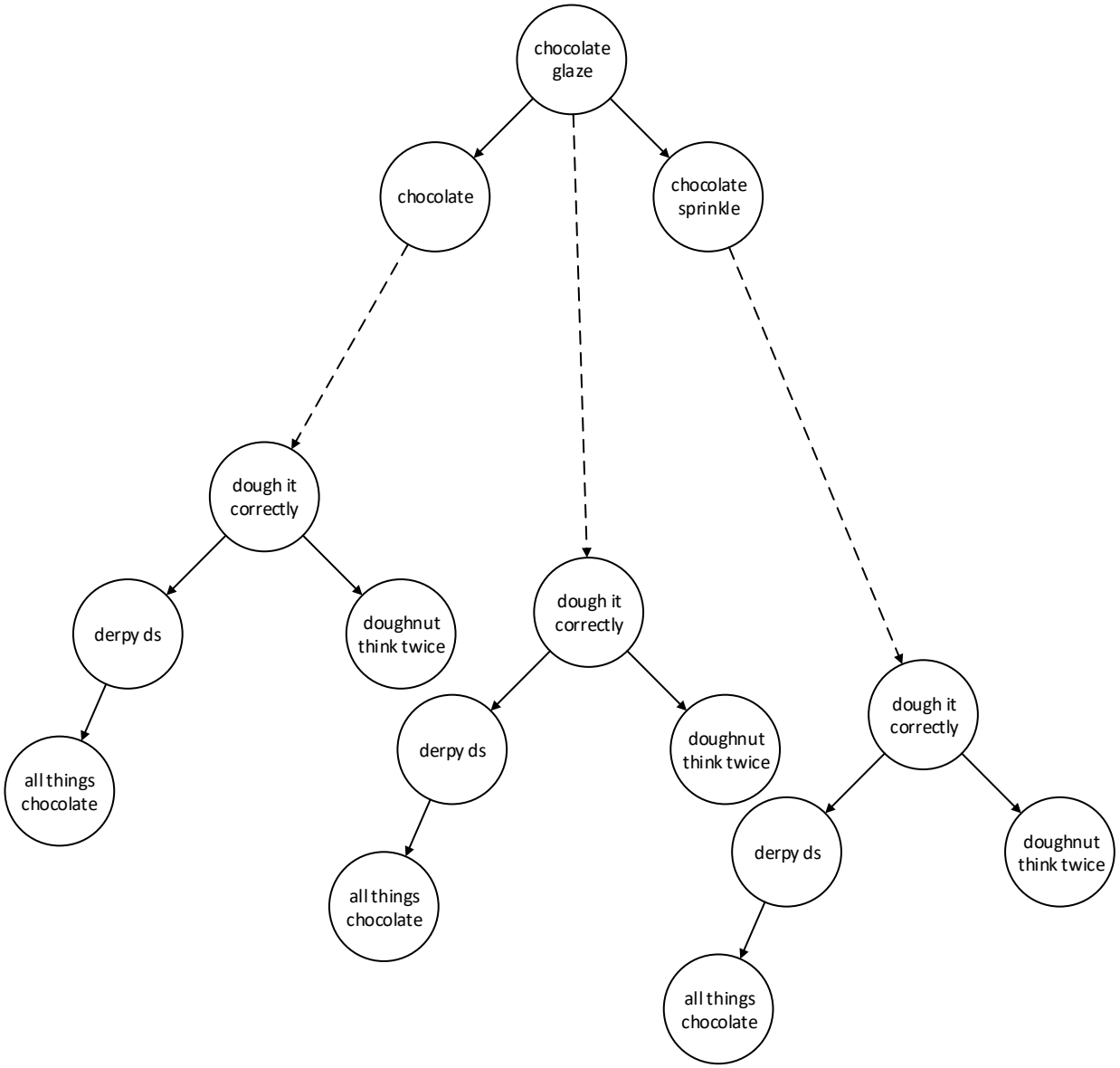
Then we add shops and doughnuts. The added doughnuts are chocolate, chocolate_glaze, and chocolate_sprinkle to the shops derpy_ds, dough_it_correctly, doughnut_think_twice, and all_things_chocolate.

The big shop tree now look like the following



Explanation continued on next page

The big doughnut tree will look like the following



We then add jelly and custard to derpy ds and doughnut do puns. The last add gives dough it correctly a jelly and a mango doughnut. The final list of shops is

Shop name	**all things chocolate**	**derpy ds**	**dough it correctly**	**doughnut do puns**	**doughnut think twice**
Goods	chocolate	chocolate	chocolate	custard	chocolate
	chocolate glaze	chocolate glaze	chocolate glaze	jelly	chocolate glaze
	chocolate sprinkle	chocolate sprinkle	chocolate sprinkle		chocolate sprinkle
		custard	jelly		
		jelly	mango		

The final list of doughnuts is

Food name	chocolate	chocolate glaze	chocolate sprinkle	custard	jelly	mango
Shops	all things chocolate	all things chocolate	all things chocolate	derpy ds	derpy ds	dough it correctly
	derpy ds	derpy ds	derpy ds	doughnut do puns	dough it correctly	
	dough it correctly	dough it correctly	dough it correctly		doughnut do puns	
	doughnut think twice	doughnut think twice	doughnut think twice			

For the first output we list all the shops that sell jelly doughnuts. The 3 shops that sell the jelly doughnuts are derpy ds, dough it correctly, and doughnut do puns.

The next section of output contains the doughnuts sold by derpy ds. The 5 doughnuts are chocolate, chocolate glaze, chocolate sprinkle, custard, and jelly.

The last section contains the doughnuts sold by dough it correctly. They sell 5 doughnuts as well: chocolate, chocolate glaze, chocolate sprinkle, custard, and mango.

Case 2

The first output requests the doughnuts sold by eat it, but eat it at the time only has 3 doughnuts.

After adding a few more doughnuts to eat it (bunt and lemon poppy) we request the updated menu.

The last section of output contains the number of doughnuts sold by the shop called does not exists. Since no doughnuts have been added to the shop, a zero is outputted.

Grading Information
Reading from standard input/output – 10 points

No extra IO – 10 points

Implements Binary Search Trees (or something to this effect e.g. hast table, trie, etc.) – 10 points

Compounded the Binary Search Trees (Trees of Trees) (or something to the effect e.g. table of trees, tree of tries, trie of tries, etc.), so that we can search for a particular shop or a particular doughnut – 10 points

Good comments/variable names/whitespace usage – 10 points

Your program will be tested on 10 test cases – 5 points each

No points will be awarded to programs that do not compile.

Solutions without an "advanced" data structure (tree, hash table, trie, etc.) will receive a maximum of 50 points

Only cases that finish within the maximum of {5 times the judge solution, 10 seconds} will be graded.