Задача Е. КОРЕНОВО ДЪРВО

Всяко ***дърво*** (т.е. свързан граф без цикли) можем да превърнем в ***кореново дърво***, като изберем един от върховете, да го означим с *r*, за ***корен***. Добре известно е, че всеки два върха на дърво са свързани с единствен път и броя на ребрата по този път наричаме ***разстояние*** между двата върха. Ако върхът *u* е на разстояние *d* от *r*, върхът *v* е на разстояние *d* + *k* от *r*, *k* > 0, и *u* е на пътя от *r* до *v*, тогава *u* се нарича ***k-ти предшественик*** на *v*, а *v* – ***наследник*** на *u*. Върховете, които нямат наследници, се наричат ***листа*** на кореновото дърво. Напишете програма, която по зададено кореново дърво изпълнява следните заявки:

- `0` *x y* – добавя в дървото нов лист *x*, като го свързва с ребро към *y*;
- `1` *x* – премахва листа *x* от дървото;
- `2` *x k* – запитване за *k*-тия предшественик на *x*.

Вход: Програмата трябва да може да обработва няколко примера при едно изпълнение. На първия ред на **стандартния вход** ще бъде зададен броят *T* на тестовите примери. Първият ред на всеки от тях ще съдържа броя *N* на върховете на зададеното кореново дърво и броя *Q* на заявките. Следват *N* реда с по две числа *x* и *y*, указващи че *x* е пряк наследник (дете) на *y*. Когато *y* е 0, това означава, че *x* е корен на дървото (т.е няма баща). Следват *Q* реда, съдържащи по един от трите типа заявки, споменати по-горе.

Изход: За всяка заявка от тип `2`, програмата трябва да изведе на **стандартния изход** *k*-тия предшественик на *x*. Ако такъв предшественик не съществува или в дървотоняма връх *x*, тогава програмата трябва да изведе 0.

Ограничения: $1 \le N \le 10^5$, $1 \le Q \le 10^5$, $1 \le x \le 10^5$, $0 \le y < 10^5$, $1 \le k \le 10^5$.

Пример:

Вход	**Изход**
1 8 11 4 0 1 4 7 1 3 2 2 4 9 1 6 2 124 9 2 124 3 1 124 2 124 3 0 55 7 0 8 55 0 10 8 2 10 4 2 6 1 2 3 2 2 9 7 2 8 3	4 0 1 2 4 0 1

Task E. ROOTED TREE

Each ***tree*** (i.e. connected graph without ccircuits) could be transformed in a ***rooted tree***, by selecting one of the vertices, to denote it by *r*, for ***root***. It is well known that every two vertices of the tree are linked with a single path and the number of edges in this path is called a ***distance*** between the two vertices. If *u* is at a distance *d* from *r*, *v* is at a distance $d + k$ from *r*, $k > 0$, and *u* is on the path from *r* to *v*, then *u* is called a ***k-th predecessor*** of *v*, and *v* – a ***successor*** of *u*. The vertices that have no successors are called ***leaves*** of the rooted tree. Write a program that, for a given rooted tree performs the following queries:

- `0` *x y* – inserts a new leaf *x* in the tree and links it with an edge to *y*;
- `1` *x* – deletes the leaf *x* from the tree;
- `2` *x k* – asks for the *k*-th predecessor of *x*.

Input: The program must be able to handle a few test cases. The first line of the **standard input** will contain one integer *T* – the number of test cases. The first line of each of them will contain the number *N* of vertices of the rooted tree and the number *Q* of the queries. Then *N* lines follow with two numbers *x* and *y*, indicating that *x* is a direct successor (child) of *y*. When *y* is 0, this means that *x* is the root of the tree (0 is not a part of the tree). Each of the following *Q* rows contains one of the three types of queries mentioned above.

Output: For each query of type 2 the program must print to the **standard output** the *k*-th predecessor of *x*. If there is no such predecessor or the tree has not vertex *x*, then the program should print 0.

Restrictions: $1 \le N \le 10^5$, $1 \le Q \le 10^5$, $1 \le x \le 10^5$, $0 \le y < 10^5$, $1 \le k \le 10^5$.

Example:

Input	Output
1 8 11 4 0 1 4 7 1 3 2 2 4 9 1 6 2 124 9 2 124 3 1 124 2 124 3 0 55 7 0 8 55 0 10 8 2 10 4 2 6 1 2 3 2 2 9 7 2 8 3	4 0 1 2 4 0 1