**10802 CPP Final Exam**

|  |
| --- |
| **Contributor： HW** |
| **Subject： Degree of kinship** |
| **Main testing concept： Pointer, Recursion, Flow of Control**   |  |  | | --- | --- | | **Basics** | **Functions** | | ■ C++ BASICS  ■ FLOW OF CONTROL  ■ FUNCTION BASICS  □ PARAMETERS AND OVERLOADING  □ ARRAYS  □ STRUCTURES AND CLASSES  □ CONSTRUCTORS AND OTHER TOOLS  □ OPERATOR OVERLOADING, FRIENDS, AND REFERENCES  □ STRINGS  ■ POINTERS AND DYNAMIC ARRAYS | □ SEPARATE COMPILATION AND NAMESPACES  □ STREAMS AND FILE I/O  ■ RECURSION  □ INHERITANCE  □ POLYMORPHISM AND VIRTUAL FUNCTIONS  □ TEMPLATES  ■ LINKED DATA STRUCTURES  □ EXCEPTION HANDLING  □ STANDARD TEMPLATE LIBRARY  □ PATTERNS AND UML | |
| **Description：**  The task is to find the degree of kinship between two people while Given several parent-child relationships. The degree of kinship measures the distance of relations between two people. To calculate the shortest kinship distance, you must locate their common ancestor, traverse from one person up to the common ancestor based on the parent-child relation, and traverse from the ancestor down to the other person. Each traverse counts for one unit of distance.    Take the figure above as an example. (1) The degree of kinship between Leo and Chris is 2 (Leo→Peter→Chris) ; (2) The degree of kinship between Leo and Mick is 3 (Leo→Peter→Sam→Mick).  Notes:   1. There are some assumptions for the test:  * Only one person is the ancestor of all people. * Each person only has one parent. * Each person can have at most two children. * All persons of a parent should be linked in ASCII order (from large to small). For example, you should have the Leo as the left child and Chris as the right child for Peter’s children, Leo and Chris.  1. You DON’T need to consider any exception. All tests ensure the existence of the relationship between two people.   **Input：**   1. The first line inputs an integer N which represents the number of relationships in the tree. 2. The following N lines input two strings, P and C, split by a space where each line record P is C’s parent. (Please check the example input and graph in the following). 3. Enter the lowest level nodes and their parent node. 4. Enter the second level nodes will be entered with his parent node 5. And so on 6. The N+2th line inputs an integer M to express the number of queries made in the following. 7. The following M lines express the quires of two strings, A and B, split by a space to query the degree of kinship between A and B.   **Output：**  There are A and B. Find each depth of them. To find their common ancestor. Now you can go through A to B by common ancestor. You should find A’s and B’s all the parents until the common ancestor. First, you should go through the parents from A to common ancestor. Then, you should find B from the common ancestor's children.   1. Output all tree nodes according to the level in the format of “Level n: XXX”. You should start from the root which is the first level. Persons in the same level should be listed based on the ASCII order (from large to small), and they should be separated by a comma, ",", and one space, " ". 2. Output the process finding name B from name A by    1. outputting each of the distance and name from A to the common ancestor.    2. outputting each of the distance and name from the common ancestor to B.   in the format of the first line of “Distance Count N” and the second line of “XXX” for where N is the accumulating distance, and XXX is the name of the node. For example, Leo and Mick then you should print out the following message.  Distance Count: 1  Peter  Distance Count: 2  Sam  Distance Count: 3  Mick   1. Output the degree of kinship for each of A and B in the format of integer. For example, Leo and Mick will output：3     **Sample Input / Output :**   |  |  |  |  | | --- | --- | --- | --- | | **Sample Input** | **Sample Output** | | | | 6  Peter Leo  Peter Chris  Mick Fob  Mick Adam  Sam Peter  Sam Mick  3  Leo Chris  Leo Mick  Leo Sam | Level 1: Sam  Level 2: Peter, Mick  Level 3: Leo, Fob, Chris, Adam  Distance Count: 1  Peter  Distance Count: 2  Chris  2  Distance Count: 1  Peter  Distance Count: 2  Sam  Distance Count: 3  Mick  3  Distance Count: 1  Peter  Distance Count: 2  Sam  2 | | | |  | | |  | |  | | |  | |
| **□ Easy, only basic programming syntax and structure are required.**  **□ Medium, multiple programming grammars and structures are required.**  **■ Hard, need to use multiple program structures or complex data types.** |
| **Expected solving time:**  60 minutes |
| **LTE:**  1Sec |
| **Other notes:** |