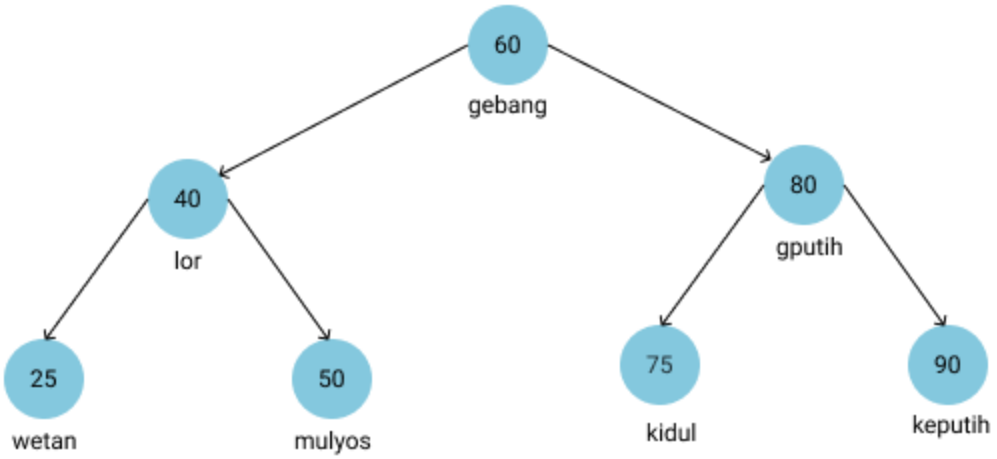
Keluarga Imron

Mata Kuliah	Struktur Data
Kelas	C
Tahun Ajaran	2021/2022
Praktikum Modul	2

Oleh: Luthfiyyah Hanifah Amari (5025201090)
Username Hackerrank: Luthfiyyah_hani1

Description

Hari ini keluarga imrom pengen healing ke negri seberang. katanya, kota di negri tersebut memiliki id dan nama. saat imron melihat peta negri tersebut, ternyata posisi kota-kota pada negri tersebut berbentuk binary tree.



*setiap node merepresentasikan kota. angka di dalamnya adalah id. tulisan di bawahnya adalah nama kota

untuk menuju suatu kota, perlu mengikuti jalur pada peta. Jadi, untuk menuju suatu kota, keluarga imron kadang perlu melewati kota-kota lainnya. Misalnya, dalam gambar diatas, untuk menuju kota dengan id 50, kita perlu melewati kota gebang dan lor.

kebetulan, imron pengen pergi ke kota yang berbeda dengan keluarganya karena ada kebutuhan lain. Rencananya, setelah urusan masing-masing selesai, imron dan keluarganya mau ketemu di kota Terdekat (dari kota tujuan imron dan keluarganya) yang sama-sama mereka lewati.

bantulah keluarga imron untuk menentukan kota mana yang menjadi kota pertemuan antara imron dan keluarganya

Input Format

baris pertama adalah n, yang merupakan jumlah kota.
n baris selanjutnya adalah id dan nama kota yang akan diinsert ke dalam tree negri tersebut sesuai dengan aturan insert pada binary tree
baris selanjutnya adalah t, yang merupakan jumlah testcase
t baris berikutnya berisi input
x y
di mana

- x merupakan id kota yg dikunjungi imron
- y merupakan id kota yg dikunjungi keluarga imron

Constraints

- $1 < n, t < 20$
- $0 < id < 1000$
- id setiap kota tidak ada yang sama
- nama setiap kota hanya terdiri dari 1 kata

Output Format

pada setiap testcase, keluarkan output berupa nama kota yang menjadi kota pertemuan antara imron dan keluarganya (kota Terdekat dari kota tujuan imron dan keluarganya yang sama-sama mereka lewati)

Sample Input 0

```
7
60 gebang
40 lor
25 wetan
50 mulyos
80 gputih
90 keputih
75 kidul
3
25 50
80 50
40 25
```

Sample Output 0

```
lor
gebang
lor
```

Pembahasan

untuk case mencari parent, memakai binary search tree. untuk case mencari index, dapat memakai binary tree atau pun struktur data lain karena testcase lemah. Apabila testcase-nya kuat (input data dalam skala besar), solusinya dapat menggunakan binary

tree. Yakni, dengan menyimpan jumlah anak kiri dari suatu node. Lalu, untuk mencari index pada suatu node, dapat menghitung '1 + jumlah anak kiri + (jumlah anak kiri dari parent + 1)'

Source Code

```
#include <stdlib.h>
#include <stdbool.h>
#include <stdio.h>

/**
 * Node structure and
 * uniqueBST structure
 */

typedef struct bstnode_t {
    int key;
    struct bstnode_t \
        *left, *right;
    char nama[100];
} BSTNode;

typedef struct bst_t {
    BSTNode *_root;
    unsigned int _size;
} BST;

/**
 * !!! WARNING UTILITY FUNCTION !!!
 * Recognized by prefix "__bst__"
 * -------------------------------------------
 * Note that you better never access these functions,
 * unless you need to modify or you know how these functions work.
 */

BSTNode* __bst__createNode(int value, char nama[]) {
    BSTNode *newNode = (BSTNode*) malloc(sizeof(BSTNode));
    newNode->key = value;
    newNode->left = newNode->right = NULL;
    strcpy(newNode->nama, nama);
    return newNode;
}

BSTNode* __bst__insert(BSTNode *root, int value, char nama[]) {
    if (root == NULL)
        return __bst__createNode(value, nama);

    if (value < root->key)
```

```
        root->left = __bst__insert(root->left, value, nama);
    else if (value > root->key)
        root->right = __bst__insert(root->right, value, nama);

    return root;
}

BSTNode* __bst__search(BSTNode *root, int value) {
    while (root != NULL) {
        if (value < root->key)
            root = root->left;
        else if (value > root->key)
            root = root->right;
        else
            return root;
    }
    return root;
}

BSTNode* __bst__findMinNode(BSTNode *node) {
    BSTNode *currNode = node;
    while (currNode && currNode->left != NULL)
        currNode = currNode->left;

    return currNode;
}

/**
 * PRIMARY FUNCTION
 * --------------------------
 * Accessible and safe to use.
 */

void bst_init(BST *bst) {
    bst->_root = NULL;
    bst->_size = 0u;
}

bool bst_isEmpty(BST *bst) {
    return bst->_root == NULL;
}

bool bst_find(BST *bst, int value) {
    BSTNode *temp = __bst__search(bst->_root, value);
    if (temp == NULL)
        return false;

    if (temp->key == value)
        return true;
    else
        return false;
}

void bst_insert(BST *bst, int value, char nama[]) {
```

```
    if (!bst_find(bst, value)) {
        bst->_root = __bst__insert(bst->_root, value, nama);
        bst->_size++;
    }
}

char* cari_kota(BST* bst, int B1, int B2){

    BSTNode* root = bst->_root;
    while(1){

        if( (B1<root->key && B2>root->key) ||(B2<root->key && B1>root->key)){
            return root->nama;
        }
        else if( B1<root->key && B2<root->key){
            root = root->left;
        }
        else if( B1>root->key && B2>root->key){
            root = root->right;
        }
        else if( B1==root->key || B2==root->key){
            return root->nama;
        }

    }

}
int main()
{
    BST bst;
    int N;
    int B; //id yang akan dimasukkan ke pohon keluarga
    int B1, B2; //2 id kota yang akan dicek kota pertemuannya
    int i;
    char nama_kota[100];
    bst_init(&bst);

    scanf("%d", &N);

    for(i=0; i<N; i++)
    {
        scanf("%d %s", &B, &nama_kota);
        bst_insert(&bst, B, nama_kota);
    }

    int t;
    scanf("%d", &t);
    for(i=0; i<t; i++)
    {
        scanf("%d %d", &B1, &B2);
        printf("%s\n", cari_kota(&bst, B1, B2));
    }
```

```
    return 0;
}
```