绝世好题

Description

想必各位身经百战的同学都已经熟练掌握了如何写一个链表，数组的使用更是不在话下。下面我们就来写一个以链表为元素的数组吧！

你需要完成一个链表数组类型 `LinkListArray`，里面包含 len 个保存正整数的单链表（你可以自由决定是否带头结点），**并要求在任意时刻每个链表里的元素都是满足非严格升序排列的（可以有存在相同的元素）**，也就是说，在任意时刻，每个单链表中的第 0 个元素应该是这个单链表中非严格最小的元素，第 1 个元素应该是这个单链表中非严格次小的元素……链表中允许元素重复。开始时，每一个单链表中都没有元素。你应该支持以下操作（所有数均为0-base）：

- 0 在第 s_1 个链表中加入数字 s_2
- 1 查询第 s_1 个链表中的第 s_2 个元素，如果第 s_1 个链表中的元素数小于 s_2，则返回 -1。
- 2 将第 s_1 个链表与第 s_2 个链表合并成一个升序链表，保存为第 s_1 个链表，将第 s_2 个链表清空 ($s_1 \neq s_2$)。

对于如何合并两个有序链表得到一个新的有序链表，你应该使用如下算法：

1、如果两个链表都为空，则算法结束。

2、如果一个链表非空，另外一个链表为空，则将非空链表的最小元素从链表中取出，加入新链表的末尾。回到第1步。

3、如果两个链表皆非空，比较两个链表各自的最小元素，将两个元素较小的那个取出，加入新链表的末尾。回到第1步。

这样就得到了新的有序链表。当然，你应该自己填补算法的细节。

然而善良的助教已经写好了完整的 `LinkListArray` 类，你只需要填补它所依赖的 `LinkList` 类就好啦！

注意事项：

- 你应该把链表中的每一个节点封装为一个类/结构体。每个节点包含一个数据元素和一个后继指针。
- 你应该把每一个单链表也封装成为一个结构体。
- 为防止内存泄漏，你需要在程序最后释放所有申请的堆空间。
- 不要修改 `LinkList` 类以外的代码。

Input Format

第一行两个数 len 和 m 表示链表数组长度和总操作数。

接下来的 m 行，每行包含第一个数 op 表示操作类型，后面紧跟两个正整数，表示题目中的 s_1, s_2。保证操作合法且所给部分和所给要求不会产生超时问题。

Output Format

对于每个操作 1，输出一行一个正整数，表示查询的结果。

Input Sample

```
4 9
0 0 5
0 0 3
1 0 0
1 0 2
0 1 4
2 1 0
1 1 1
1 1 0
1 0 0
```

Output Sample

```
3
-1
4
3
-1
```

Hint

对于 30% 的数据，$len = 1$。

对于 60% 的数据，不包含操作2。

对于 100% 的数据，$1 \leq len \leq 100$，数组中元素小于等于10^9。

- 使用与题目要求不同的方法或没有使用Sample Code代码通过测试点，会扣除相应测试点的分数
- 存在内存泄漏会额外扣除20分
- 同时，此题需要你注意提交代码的代码风格！！！极为糟糕的代码风格（例如改变语法的某些宏定义）会酌情扣分！！！

本题只有一次提交机会！！！

Sample Code

```
#include <bits/stdc++.h>

class LinkList {
    //TODO
};

class LinkListArray {
private:
    int len;
    LinkList **lists;

public:
    LinkListArray(int n): len(n) {
        lists = new LinkList*[n];
        for (int i = 0; i < n; ++i) lists[i] = new LinkList;
    }
    ~LinkListArray() {
        for (int i = 0; i < len; ++i) {
            delete lists[i];
        }
        delete []lists;
    }

    void insertNumber(int n, int x) {
        lists[n]->push(x);
    }
    int queryNumber(int n, int k) {
        return lists[n]->getKth(k);
    }
    void mergeLists(int p, int q) {
        lists[p]->merge(lists[q]);
    }
};

int main() {
    int len, m;
    scanf("%d%d", &len, &m);
    LinkListArray a = LinkListArray(len);
    for (int i = 0, op, s1, s2; i < m; ++i) {
        scanf("%d%d%d", &op, &s1, &s2);
        if (op == 0) {
            a.insertNumber(s1, s2);
        }
        if (op == 1) {
            printf("%d\n", a.queryNumber(s1, s2));
        }
```

```
        if (op == 2) {
            a.mergeLists(s1, s2);
        }
    }
    return 0;
}
```