**问题描述**

|  |  |
| --- | --- |
| 试题编号： | 201709-2 |
| 试题名称： | 公共钥匙盒 |
| 时间限制： | 1.0s |
| 内存限制： | 256.0MB |
| 问题描述： | **问题描述**  　　有一个学校的老师共用*N*个教室，按照规定，所有的钥匙都必须放在公共钥匙盒里，老师不能带钥匙回家。每次老师上课前，都从公共钥匙盒里找到自己上课的教室的钥匙去开门，上完课后，再将钥匙放回到钥匙盒中。 　　钥匙盒一共有*N*个挂钩，从左到右排成一排，用来挂*N*个教室的钥匙。一串钥匙没有固定的悬挂位置，但钥匙上有标识，所以老师们不会弄混钥匙。 　　每次取钥匙的时候，老师们都会找到自己所需要的钥匙将其取走，而不会移动其他钥匙。每次还钥匙的时候，还钥匙的老师会找到最左边的空的挂钩，将钥匙挂在这个挂钩上。如果有多位老师还钥匙，则他们按钥匙编号从小到大的顺序还。如果同一时刻既有老师还钥匙又有老师取钥匙，则老师们会先将钥匙全还回去再取出。 　　今天开始的时候钥匙是按编号从小到大的顺序放在钥匙盒里的。有*K*位老师要上课，给出每位老师所需要的钥匙、开始上课的时间和上课的时长，假设下课时间就是还钥匙时间，请问最终钥匙盒里面钥匙的顺序是怎样的？  **输入格式**  　　输入的第一行包含两个整数*N*, *K*。 　　接下来*K*行，每行三个整数*w*, *s*, *c*，分别表示一位老师要使用的钥匙编号、开始上课的时间和上课的时长。可能有多位老师使用同一把钥匙，但是老师使用钥匙的时间不会重叠。 　　保证输入数据满足输入格式，你不用检查数据合法性。  **输出格式**  　　输出一行，包含*N*个整数，相邻整数间用一个空格分隔，依次表示每个挂钩上挂的钥匙编号。  **样例输入**  5 2 4 3 3 2 2 7  **样例输出**  1 4 3 2 5  **样例说明**  　　第一位老师从时刻3开始使用4号教室的钥匙，使用3单位时间，所以在时刻6还钥匙。第二位老师从时刻2开始使用钥匙，使用7单位时间，所以在时刻9还钥匙。 　　每个关键时刻后的钥匙状态如下（X表示空）： 　　时刻2后为1X345； 　　时刻3后为1X3X5； 　　时刻6后为143X5； 　　时刻9后为14325。  **样例输入**  5 7 1 1 14 3 3 12 1 15 12 2 7 20 3 18 12 4 21 19 5 30 9  **样例输出**  1 2 3 5 4  **评测用例规模与约定**  　　对于30%的评测用例，1 ≤ *N*, *K* ≤ 10, 1 ≤ *w* ≤ *N*, 1 ≤ *s*, *c* ≤ 30； 　　对于60%的评测用例，1 ≤ *N*, *K* ≤ 50，1 ≤ *w* ≤ *N*，1 ≤ *s* ≤ 300，1 ≤ *c* ≤ 50； 　　对于所有评测用例，1 ≤ *N*, *K* ≤ 1000，1 ≤ *w* ≤ *N*，1 ≤ *s* ≤ 10000，1 ≤ *c* ≤ 100。 |
| 代码程序 | **#include <stdio.h>**  **#include <vector>**  **#include<algorithm>**  **using namespace std;**  **typedef struct Node**  **{**  **int keyId;**  **int time;**  **int flag; // 0 表示 借钥匙， 1 表示 还钥匙**  **}Node;**  **int Key[1005];**  **bool cmp(Node a, Node b);**  **void Deal(Node node, int N);**  **int main()**  **{**  **int N, K;**  **scanf("%d %d", &N,&K);**  **// 初始化 钥匙**  **for(int i = 1; i<= N;i++)**  **{**  **Key[i] = i;**  **}**  **Node temp;**  **int tid, jtime,htime;**  **vector<Node> vt;**  **for(int i = 1; i <= K; i++)**  **{**  **scanf("%d %d %d", &tid, &jtime, &htime);**  **htime += jtime;**  **temp.keyId = tid;**  **temp.time = jtime;**  **temp.flag = 0;**  **vt.push\_back(temp);**  **temp.time = htime;**  **temp.flag = 1;**  **vt.push\_back(temp);**  **}**  **sort(vt.begin(), vt.end(), cmp);**  **for(int i = 0; i< vt.size();i++)**  **{**  **Deal(vt[i], N);**  **}**  **for(int i = 1; i<= N;i++)**  **{**  **printf("%d", Key[i]);**  **if(i<N)**  **{**  **printf(" ");**  **}**  **}**  **return 0;**  **}**  **bool cmp(Node a, Node b)**  **{**  **if(a.time != b.time)**  **{**  **return a.time < b.time;**  **}else if(a.flag != b.flag){**  **return a.flag > b.flag;**  **}else{**  **return a.keyId < b.keyId;**  **}**  **}**  **void Deal(Node node, int N)**  **{**  **int key = node.keyId;**  **if(node.flag == 0) // 借钥匙**  **{**  **for(int i = 1; i <= N; i++)**  **{**  **if(Key[i] == key) // 找到**  **{**  **Key[i] = -1;**  **break;**  **}**  **}**  **}else{ // 还钥匙**  **for(int i = 1; i <= N; i++)**  **{**  **if(Key[i] == -1) // 找到**  **{**  **Key[i] = key;**  **break;**  **}**  **}**  **}**  **}** |