Started on	Tuesday, 1 July 2025, 10:39 PM
State	Finished
Completed on	Tuesday, 1 July 2025, 10:40 PM
Time taken	24 secs
Marks	1.00/1.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Peter rất thích chơi đá. Anh ta thường dùng đá để trang trí sân nhà của mình. Hiện tại Peter có n hòn đá. Dĩ nhiên mỗi hòn đá có một khối lượng nào đó. Peter muốn đặt các hòn đá này dọc theo lối đi từ cổng vào nhà của mình. Peter lại muốn sắp xếp như thế này: hòn đá nặng nhất sẽ đặt ở cạnh cổng rào, kế tiếp là hòn đá nặng thứ 2, ... hòn đá nhẹ nhất sẽ được đặt cạnh nhà. Như vậy nếu đi từ trong nhà ra cổng, ta sẽ gặp các hòn đá có khối lượng tăng dần.

Tuy nhiên, điều khó khăn đối với Peter là anh chỉ có một cây cân đĩa mà không có quả cân nào. Nói cách khác, mỗi lần cân Peter chỉ có thể biết được hòn đá nào nhẹ hơn hòn đá nào chứ không biết nó nặng bao nhiêu kg.

Sau m lần cân, Peter biết được sự khác nhau về cân nặng của m cặp. Với các thông tin này, hãy giúp Peter sắp xếp các viên đá theo thứ tự anh mong muốn.

Đầu vào (Input)

Dữ liệu đầu vào được nhập từ bàn phím với định dạng:

- Dòng đầu tiên chứa 2 số nguyên n và m, tương ứng là số hòn đó và số lần cân
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên u v nói rằng hòn đá u nhẹ hơn hòn đá v. Không có cặp u, v nào được lặp lại.

Đầu ra (Output)

- In ra màn hình thứ tự của các hòn đá theo khối lượng tăng dần, mỗi số trên 1 dòng.

Dữ liệu đầu vào được đảm bảo chỉ có một kết quả quả duy nhất.

Xem thêm ví dụ bên dưới. Trong ví dụ đầu tiên ta có: hòn đá 1 nhẹ nhất, kế đến là hòn đá 3 và sau cùng là hòn đá 2.

For example:

Input	Result
3 2 1 3 3 2	1 3 2
7 13 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7 2 4 6 5 4 6	1 2 4 6 5 3 7

Answer: (penalty regime: 10, 20, ... %)

```c
#include <stdio.h>

#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} Graph;


void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
        for (int v = 1; v <= n; v++)
            pG->A[u][v] = 0;
}

void add_edge(Graph *pG, int u, int v) {
    pG->A[u][v] += 1;
}


```

	Input	Expected	Got	
✔	3 2 1 3 3 2	1 3 2	1 3 2	✔
✔	7 13 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7 2 4 6 5 4 6	1 2 4 6 5 3 7	1 2 4 6 5 3 7	✔
✔	7 13 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4 6 5 5 4	1 3 2 6 5 4 7	1 3 2 6 5 4 7	✔

Passed all tests! ✔

Question author's solution (C):

```c
#include <stdio.h>

#define MAX_N 100
typedef struct {
    int n, m;
    int A[MAX_N][MAX_N];
} Graph;


void init_graph(Graph *pG, int n) {
    pG->n = n;
    pG->m = 0;
    for (int u = 1; u <= n; u++)
        for (int v = 1; v <= n; v++)
            pG->A[u][v] = 0;
}

void add_edge(Graph *pG, int u, int v) {
    pG->A[u][v] += 1;
}


```

Correct

Marks for this submission: 1.00/1.00.

◄ 006. Quản lý dự án phần mềm

Jump to...

Tổ chức thi công - Dự án xây nhà ►