Started on	Tuesday, 1 July 2025, 10:21 PM
State	Finished
Completed on	Tuesday, 1 July 2025, 10:36 PM
Time taken	14 mins 43 secs
Marks	2.00/2.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Cô giáo Trang chuẩn bị kẹo để phát cho các bé mà cô đang giữ. Dĩ nhiên môi bé đều có một tên gọi rất dễ thương ví dụ: Mạnh Phát, Diễm Quỳnh, Đăng Khoa, ... Tuy nhiên, để đơn giản vấn đề ta có thể giả sử các em được đánh số từ 1 đến n.

Cô giáo muốn rằng tất cả các em đều phải có kẹo. Cô lại biết thêm rằng có một số bé có ý muốn hơn bạn mình một chút vì thế các em ấy muốn kẹo của mình nhiều hơn của bạn.

Hãy viết chương trình giúp cô tính xem mỗi em cần được chia ít nhất bao nhiêu kẹo và tổng số kẹo ít nhất mà cô phải chuẩn bị là bao nhiêu.

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 bé và số cặp bé mà trong đó có 1 bé muốn có kẹo hơn bạn mình.
- m dòng tiếp theo mỗi dòng chứa 2 số nguyên a, b nói rằng bé a muốn có kẹo nhiều hơn bé b.

Đầu ra (Output)

- In ra màn hình số kẹo ít nhất của từng em, mỗi em trên một dòng.
- Dòng cuối cùng in tổng số kẹo ít nhất mà cô giáo Trang cần phải chuẩn bị.

Xem thêm các ví dụ để hiểu thêm về đầu vào và đầu ra.

For example:

Input	Result
3 2 1 3 3 2	3 1 2 6

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

```c
#include <stdio.h>
#define MAX_N 100

//__________LIST___________
typedef struct
{
    int data[MAX_N]; // mảng bao gồm các phần tử của danh sách
    int size;        // độ dài của danh sách
} List;

void make_null_list(List *pL)
{
    pL->size = 0;
    // (*L).size=0;
}

// Thêm một phần tử mới vào cuối ds
void push_back(List *pL, int x)
{

    // pL->data[pL->size] = x;
    // pL->size++;
```

	Input	Expected	Got	
✔	3 2 1 3 3 2	3 1 2 6	3 1 2 6	✔
✔	7 10 2 1 3 1 4 1 3 2 6 2 7 3 5 4 3 5 7 5 7 6	1 2 4 2 3 3 5 20	1 2 4 2 3 3 5 20	✔

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.

Question **2**

Correct

Mark 1.00 out of 1.00

Trong một giải đấu bóng đá gồm n đội bóng, đánh số từ 1 đến n. Mỗi trận đấu có hai đội thi đấu với nhau cho đến khi phân biệt thắng thua (ví dụ: hiệp phụ, đá luân lưu). Sau khi giải đấu kết thúc, ban tổ chức muốn xếp hạng các đội theo quy tắc sau:

- Hạng được tính từ 1, 2, 3, ...
- Đội không thua trận nào xếp hạng 1
- Nếu đội A đá **thắng** đội B thì hạng của đội A **nhỏ hơn** hạng của đội B.
- Nếu một đội có thể nhận nhiều hạng khác nhau thì chọn hạng nhỏ nhất.

Hoặc bạn cũng có thể sử dụng định nghĩa sau:

Hạng(v) = max {Hạng(u) } + 1

với u là đội thắng v.

Hãy giúp ban tổ chức viết chương trình xếp hạng cho các đội. Giả sử không xảy ra trường hợp A thắng B, B thắng C ,..., Z thắng A.

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 đội và số trận đấu.
- **m** dòng tiếp theo mỗi dòng chứa 2 số nguyên **u v** mô tả kết quả trận đấu: **u** thắng, **v** thua.

Đầu ra (Output):

- In ra màn hình hạng của các đội bóng theo số thứ tự của đội trên cùng 1 dòng, mỗi đội cách nhau 1 khoảng trắng.

<Hạng đội 1> <Hạng đội 2> ... <Hạng đội n>

Xem thêm ví dụ bên dưới. Trong ví dụ đầu tiên ta có: hạng của 1 = 1, hạng của 2 = 3 và hạng của 3 = 2.

Trong ví dụ 2: đội 2 có thể nhận hạng 2 hoặc 3 nên xếp hạng 2, tương tự như thế cho đội 4.

For example:

Input	Result
3 2 1 3 3 2	1 3 2
7 10 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7	1 2 4 2 3 3 5

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

```
#include <stdio.h>
#define MAX_N 100

//__________LIST___________
typedef struct
{
    int data[MAX_N]; // mảng bao gồm các phần tử của danh sách
    int size;        // độ dài của danh sách
} List;

void make_null_list(List *pL)
{
    pL->size = 0;
    // (*L).size=0;
}

// Thêm một phần tử mới vào cuối ds
void push_back(List *pL, int x)
{

    // pL->data[pL->size] = x;
    // pL->size++;
```

	Input	Expected	Got	
✔	3 2 1 3 3 2	1 3 2	1 3 2	✔
✔	7 10 1 2 1 3 1 4 2 3 2 6 3 7 4 5 5 3 5 7 6 7	1 2 4 2 3 3 5	1 2 4 2 3 3 5	✔
✔	7 12 1 2 1 3 2 4 2 5 2 6 3 2 3 5 3 6 4 7 5 7 6 4 6 5	1 3 2 5 5 4 6	1 3 2 5 5 4 6	✔

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.

◄ 003. Xếp hạng các đỉnh

Jump to...

005. Quản lý dự án ►