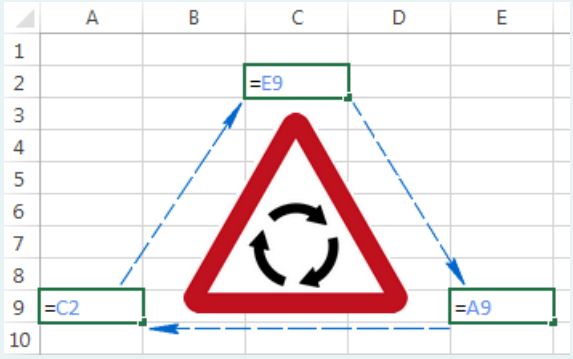
Started on	Sunday, 15 June 2025, 7:51 PM
State	Finished
Completed on	Sunday, 15 June 2025, 10:40 PM
Time taken	2 hours 48 mins
Marks	2.00/2.00
Grade	**10.00** out of 10.00 (**100**%)

Question **1**

Correct

Mark 1.00 out of 1.00

Microsoft Excel là chương trình xử lý bảng tính nằm trong bộ Microsoft Office của hãng phần mềm Microsoft. Excel được thiết kế để giúp ghi lại, trình bày các thông tin xử lý dưới dạng bảng, thực hiện tính toán và xây dựng các số liệu thống kê trực quan.



Bảng tính Excel được chia thành nhiều hàng và cột. Giao điểm của hàng và cột gọi là một ô. Trong một ô của Excel, ta có thể nhập một giá trị hoặc một công thức có sử dụng giá trị của 1 hay nhiều ô khác. Khi ô A1 sử dụng giá trị của ô B2 ta nói A1 **tham chiếu đến** B2 hay A1 **phụ thuộc vào** B2.

Một lỗi thường gặp trong Excel là **tham chiếu vòng** (a circular reference). Tham chiếu vòng là trường hợp một ô tham chiếu đến chính nó hoặc như ví dụ bên dưới: A9 tham chiếu đến C2, C2 tham chiếu đến E9 và E9 tham chiếu đến A9.

Vấn đề đặt ra là làm thế nào để phát hiện có tham chiếu vòng trong bảng tính. Giả sử bảng tính có n ô, để đơn giản, ta đánh số các ô đang xét là 1, 2, 3, ..., n. Bạn biết được ô nào tham chiếu đến (các) ô nào. Hãy viết chương trình kiểm tra xem có xuất hiện tham chiếu vòng hay không.

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 ô và số tham chiếu.
- **m** dòng tiếp theo mỗi dòng chứa 2 số nguyên **u**, **v** nói rằng ô **u** tham chiếu đến ô **v**.

Đầu ra (Output)

In ra màn hình **CIRCULAR REFERENCE** nếu bảng tính có tham chiếu vòng, ngược lại in ra **OK.**

Chú ý

- Để chạy thử chương trình, bạn nên tạo một tập tin **dt.txt** chứa đồ thị cần kiểm tra.
- Thêm dòng **freopen("dt.txt", "r", stdin);** vào ngay sau hàm main(). Khi nộp bài, nhớ gỡ bỏ dòng này ra.
- Có thể sử dụng đoạn chương trình đọc dữ liệu mẫu sau đây:

```
freopen("dt.txt", "r", stdin); //Khi nộp bài nhớ bỏ dòng này.
Graph G;
int n, m, u, v, e;
scanf("%d%d", &n, &m);
init_graph(&G, n);

for (e = 0; e < m; e++) {
    scanf("%d%d", &u, &v);
    add_edge(&G, u, v, w);
}
```

For example:

Input	Result
3 3 1 3 2 1 3 2	CIRCULAR REFERENCE
7 9 1 2 1 3 1 4 2 3 2 6 3 5 3 7 4 5 5 7	OK

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

```
1 #include <stdio.h>
```

```c
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} 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;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max];// lưu trạng thái cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    //1. tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. xét các đỉnh kề của u
    for(int v = 1; v <= pG->n; v++)
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE)  // 2a. nếu v chưa duyệt
                DFS(pG,v);         // gọi đề quy duyệt nó
            else if(color[v] == GRAY)// 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    // 3. tô màu đã duyệt xong cho u;
    color[u] = BLACK;
}
int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);

    for (int e = 0; e < m; e++)
    {
        scanf("%d%d", &u, &v);
        add_edge(&G, u, v);
    }
    //1. khoi taoj mang color[u] = WHITE
    for(int u = 1; u <= G.n; u++)
        color[u] = WHITE;

    //2. khoi tao bien has_cricle
    has_circle = 0;

    //3. duyet toan bo do thi de kiem tra chu chinh
```

```
    for(int u = 1; u <= G.n ; u++)
        if(color[u] == WHITE)
            DFS(&G,u);
    if(has_circle == 1)
        printf("CIRCULAR REFERENCE");
    else
        printf("OK");
    return 0;
}
```

	Input	Expected	Got	
✔	3 3 1 3 2 1 3 2	CIRCULAR REFERENCE	CIRCULAR REFERENCE	✔
✔	7 9 1 2 1 3 1 4 2 3 2 6 3 5 3 7 4 5 5 7	OK	OK	✔
✔	7 9 1 2 1 3 2 4 2 5 2 6 3 5 3 6 4 7 5 7	OK	OK	✔
✔	3 3 1 2 2 3 3 1	CIRCULAR REFERENCE	CIRCULAR REFERENCE	✔
✔	4 3 1 2 2 3 2 4	OK	OK	✔
✔	7 9 4 5 1 2 1 4 2 3 2 6 7 3 3 1 5 7 7 6	CIRCULAR REFERENCE	CIRCULAR REFERENCE	✔
✔	5 5 1 2 2 3 3 4 4 5 5 1	CIRCULAR REFERENCE	CIRCULAR REFERENCE	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

Question **2**

Correct

Mark 1.00 out of 1.00

Thuyền trưởng Haddock (truyện Tintin) là một người luôn say xỉn. Vì thế đôi khi Tintin không biết ông ta đang say hay tỉnh. Một ngày nọ, Tintin hỏi ông ta về cách uống. Haddock nói như thế này: Có nhiều loại thức uống (soda, wine, water, ...), tuy nhiên Haddock lại tuân theo quy tắc "*để uống một loại thức uống nào đó cần phải uống tất cả các loại thức uống tiên quyết của nó*". Ví dụ: để uống rượu (wine), Haddock cần phải uống soda và nước (water) trước. Vì thế muốn say cũng không phải dễ!

Cho danh sách các thức uống và các thức uống tiên quyết của nó. Hãy xét xem Haddock có thể nào say không ? Để làm cho Haddock say, ông ta phải uống hết tất cả các thức uống.

Ví dụ 1:

```
soda wine
water wine
```

Thức uống tiên quyết được cho dưới dạng a b, có nghĩa là để uống b bạn phải uống a trước. Trong ví dụ trên để uống wine, Haddock phải uống soda và water trước. Soda và water không có thức uống tiên quyết nên Haddock sẽ SAY.

Ví dụ 2:

```
soda wine
water wine
wine water
```

Để uống wine, cần uống water. Tuy nhiên để uống water lại cần wine. Vì thế Haddock không thể uống hết được các thức uống nên ông ta KHÔNG SAY.

Để đơn giản ta có thể giả sử các thức uống được mã hoá thành các số nguyên từ 1, 2, ... và dữ liệu đầu vào được cho có dạng như sau (ví dụ 1):

```
3 2
1 2
3 2
```

Có loại thức uống (soda: 1, wine: 2 và water: 3) và có 2 điều kiện tiên quyết

1 -> 2 và 3 -> 2.

Với ví dụ 2, ta có dữ liệu:

```
3 3
1 2
3 2
2 3
```

Viết chương trình đọc dữ liệu các thức uống và kiểm tra xem Haddock có thể say không. Nếu có in ra "YES", ngược lại in ra "NO".

Đầu vào (Input):

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

- Dòng đầu tiên chứa 2 số nguyên **n** và **m**, tương ứng là số thức uống và số điều kiện tiên quyết
- **m** dòng tiếp theo mỗi dòng chứa 2 số nguyên **u v** nói rằng thức uống **u** là tiên quyết của thức uống **v**

Đầu ra (Output):

- Nếu Haddock có thể say in ra YES, ngược lại in ra NO.

Xem thêm các ví dụ bên dưới.

For example:

Input	Result
3 2 1 2 3 2	YES
3 3 1 2 3 2 2 3	NO

Input	Result
5 5 1 2 2 3 3 4 4 2 5 4	NO

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

```c
#include <stdio.h>
#define max 100
typedef int ElementType;

typedef struct
{
    int n, m;
    int A[max][max];
} 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;
    // pG->A[v][u] = 1;
    pG->m++;
}

int adjacent(Graph *pG, int u, int v)
{
    return pG->A[u][v] > 0;
}

/* kiểm tra đồ thị có chứa chu trình*/
#define WHITE 0
#define GRAY 1
#define BLACK 2

int color[max];// lưu trạng thái cùa các đỉnh
int has_circle;// đồ thị chứa chu trình hay không

void DFS(Graph *pG, int u){
    //1. tô màu đang duyệt cho u
    color[u] = GRAY;

    //2. xét các đỉnh kề của u
    for(int v = 1; v <= pG->n; v++)
        if(adjacent(pG, u,v)){
            if(color[v] == WHITE)  // 2a. nếu v chưa duyệt
                DFS(pG,v);         // gọi đề quy duyệt nó
            else if(color[v] == GRAY)// 2b. nếu v đang duyệt
                has_circle = 1; //chứa chu trình
        }

    // 3. tô màu đã duyệt xong cho u;
    color[u] = BLACK;
}
int main()
{
    Graph G;
    int n, m, u, v;
    scanf("%d%d", &n, &m);
    init_graph(&G, n);
```

```
66
67      for (int e = 0; e < m; e++)
68      {
69          scanf("%d%d", &u, &v);
70          add_edge(&G, u, v);
71      }
72      //1. khoi taoj mang color[u] = WHITE
73      for(int u = 1; u <= G.n; u++)
74          color[u] = WHITE;
75
76      //2. khoi tao bien has_cricle
77      has_circle = 0;
78
79      //3. duyet toan bo do thi de kiem tra chu chinh
80      for(int u = 1; u <= G.n ; u++)
81          if(color[u] == WHITE)
82              DFS(&G,u);
83      if(has_circle == 1)
84          printf("NO");// có chu trình thì sẽ ko say
85      else
86          printf("YES");// ko có chu trình thì sẽ say
87      return 0;
88  }
89
```

	Input	Expected	Got	
✔	3 2 1 2 3 2	YES	YES	✔
✔	3 3 1 2 3 2 2 3	NO	NO	✔
✔	5 5 1 2 2 3 3 4 4 2 5 4	NO	NO	✔
✔	3 3 1 3 2 1 3 2	NO	NO	✔

Passed all tests! ✔

Correct

Marks for this submission: 1.00/1.00.

◄ * Bài tập 10. Kiểm tra đồ thị chứa chu trình

Jump to...

* Bài tập 12 - Kiểm tra đồ thị phân đôi ►