```c
// Xep hang do thi

#include <stdio.h>
#define MAX_VERTICES 100

// List
typedef struct {
    int data[MAX_VERTICES];
    int size;
} List;

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

void push_back(List* L, int x) {
    L->data[L->size] = x;
    ++L->size;
}

int element_at(List* L, int i) {
    return L->data[i - 1];
}

void copy_list(List* s1, List* s2) {
    make_null_list(s1);

    int i;
    for (i = 1; i <= s2->size; ++i) {
        push_back(s1, element_at(s2, i));
    }
}

// Graph
typedef struct {
    int A[MAX_VERTICES][MAX_VERTICES];
    int n;
} Graph;

void init_graph(Graph* G, int n) {
    G->n = n;

    int i, j;
```

```
    for (i = 1; i <= n; ++i) {
        for (j = 1; j <= n; ++j) {
            G->A[i][j] = 0;
        }
    }
}

void add_edge(Graph* G, int x, int y) {
    G->A[x][y] = 1;
}

int adjacent(Graph* G, int x, int y) {
    return G->A[x][y];
}

int rank[MAX_VERTICES];

void ranking(Graph* G) {
    int d[MAX_VERTICES];
    int x, u;

    for (u = 1; u <= G->n; ++u) {
        d[u] = 0;
    }

    for (x = 1; x <= G->n; ++x) {
        for (u = 1; u <= G->n; ++u) {
            if (adjacent(G, x, u)) {
                ++d[u];
            }
        }
    }

    List s1, s2;

    make_null_list(&s1);

    for (u = 1; u <= G->n; ++u) {
        if (!d[u]) {
            push_back(&s1, u);
        }
    }
```

```
    int k = 0, i;

    while (s1.size) {
        make_null_list(&s2);

        for (i = 1; i <= s1.size; ++i) {
            int u = element_at(&s1, i);

            rank[u] = k;

            int v;

            for (v = 1; v <=G->n; ++v) {
                if (adjacent(G, u, v)) {
                    --d[v];

                    if (!d[v]) {
                        push_back(&s2, v);
                    }
                }
            }
        }

        copy_list(&s1, &s2);

        ++k;
    }
}

int main() {
    Graph G;
    int n, m, u, v, w, 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);
    }

    ranking(&G);
```

```
    int i;
    for (i = 1; i <= n; ++i) {
        printf("%d \n", rank[i]);
    }

    return 0;
}
```