

```
// Duyệt do thi theo chieu Sau su dung Stack  
#include <stdio.h>
```

```
int mark[100];
```

```
// List
```

```
typedef struct {  
    int data[100];  
    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];  
}
```

```
// Stack
```

```
typedef List Stack;
```

```
void make_null_stack(Stack* S) {  
    S->size = 0;  
}
```

```
void push(Stack* S, int x) {  
    S->data[S->size] = x;  
    ++S->size;  
}
```

```
int top(Stack* S) {  
    return S->data[S->size - 1];  
}
```

```
void pop(Stack* S) {  
    --S->size;  
}
```

```

int empty(Stack* S) {
    return S->size == 0;
}

// Graph
typedef struct {
    int A[100][100];
    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_egde(Graph* G, int x, int y) {
    G->A[x][y] = 1;
    G->A[y][x] = 1;
}

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

List neighbors(Graph* G, int x) {
    int y;
    List list;

    make_null_list(&list);

    for (y = 1; y <= G->n; ++y) {
        if (adjacent(G, x, y)) {
            push_back(&list, y);
        }
    }
}

```

```

        return list;
    }

void depth_first_search(Graph* G, int start) {
    Stack frontier;
    make_null_stack(&frontier);

    push(&frontier, start);

    while (!empty(&frontier)) {
        int x = top(&frontier);
        pop(&frontier);

        if (!mark[x]) {
            printf("%d\n", x);
        }
        mark[x] = 1;

        List list = neighbors(G, x);

        int j;
        for (j = 1; j <= list.size; ++j) {
            int y = element_at(&list, j);

            if (!mark[y]) {
                push(&frontier, y);
            }
        }
    }
}

int main() {
    //freopen("dt.txt", "r", stdin);

    Graph G;
    int n, m, i, x, y;

    scanf("%d%d", &n, &m);

    init_graph(&G, n);

    for (i = 1; i <= m; ++i) {

```

```
scanf("%d%d", &x, &y);

add_egde(&G, x, y);
}

for (i = 1; i <= n; ++i) {
    mark[i] = 0;
}

for (i = 1; i <= n; ++i) {
    if (!mark[i]) {
        depth_first_search(&G, i);
    }
}

return 0;
}
```