```c
// Duyet do thi
// Duyet theo chieu Sau su dung hang doi(queue)
#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];
}

// Queue
typedef struct {
    int data[100];
    int front, rear;
} Queue;

void make_null_queue(Queue* Q) {
    Q->front = 0;
    Q->rear = -1;
}

void push(Queue* Q, int x) {
    ++Q->rear;
    Q->data[Q->rear] = x;
}

int top(Queue* Q) {
    return Q->data[Q->front];
}
```

```
void pop(Queue* Q) {
    ++Q->front;
}

int empty(Queue* Q) {
    return Q->front > Q->rear;
}

// 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);
        }
    }
```

```c
    return list;
}

void breath_first_search(Graph* G, int start) {
    Queue frontier;
    make_null_queue(&frontier);

    push(&frontier, start);
    mark[start] = 1;

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

        printf("%d\n", x);

        List list = neighbors(G, x);

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

            if (!mark[y]) {
                mark[y] = 1;
                push(&frontier, y);
            }
        }
    }
}

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

    Graph G;
    int n, m, i, j, 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]) {
            breath_first_search(&G, i);
        }
    }

    return 0;
}
```