```c
#include <stdio.h>
#define MAX_Length 20

typedef struct
{
    /* data */
    int u, v; // u = dinh 1, v = dinh 2
}edge;

typedef struct
{
    /* data */
    int n, m; // n = so dinh, m = so canh
    edge data[MAX_Length]; // e = tap hop cac canh
}graph;

typedef struct
{
    int data[MAX_Length];
    int size;
}list;

void makeNull(list *L){
    L->size = 0;
}
int elementAt(list *L, int x){
    return L->data[x];
}
void pushList(list *L, int x){
    L->data[L->size] = x;
    L->size++;
}

void initGraph(graph *G, int n){
    G->n = n;
    G->m = 0;
    printf("Do thi G duoc khoi tao voi so dinh n = %d va so canh m = %d\n", G->n, G->m);
}

void addEdge(graph *G, int u, int v){
    int i;
    for(i = 0; i<G->m; i++){
        if ((G->data[i].u == u && G->data[i].v == v) || (G->data[i].u == v && G->data[i].v == u))
        {
            printf("Canh da co trong do thi!");
            return;
        }
    }
    G->data[G->m].u = u;
    G->data[G->m].v = v;
    G->m++;
    printf("Do thi da them vao canh %d %d\n", G->data[G->m-1].u, G->data[G->m-1].v);
}
int adjacent(graph *G, int u, int v){
    int i;
    for(i = 0; i<=G->m; i++){
        if ((G->data[i].u == u && G->data[i].v == v) || (G->data[i].v == u && G->data[i].u == v))
        {
            return 1;
        }
    }
    return 0;
}

void degree(graph *G, int x){
    int i, count = 0;
    for(i = 0; i<=G->m; i++){
        if (adjacent(G, x, i))
        {
            count++;
        }
    }
    printf("Bac cua canh %d la: %d\n", x, count);
}
```

```
list neighbors(graph *G, int x){
    list listVertex;
    makeNull(&listVertex);
    int i;
    for(i = 1; i<=G->n; i++){
        if (adjacent(G, x, i) == 1)
        {
            pushList(&listVertex, i);
        }
    }
    return listVertex;
}

// Stack

typedef struct {
    int data[100];
    int size;
}Stack;

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

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

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

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

int empty(Stack S) {
    return S.size == 0;
}


// DUYET THEO CHIEU SAU

void depth_first_search(graph *G) {
    printf("\nDuyet DFS:\n");
    Stack L;
    int mark[100];
    make_null_stack(&L);

    int i;
    for (i = 1; i<= G->n; i++)
        mark[i] = 0;


    for(i = 1; i<=G->n; i++) {
        if(mark[i] == 0) {
            push(&L, i);
            while(!empty(L)) {
                int x = top(L);
                pop(&L);
                if(mark[x] != 0) continue;
                printf("Duyet %d\n",x );
                mark[x] =1;
                list ls = neighbors(G, x);
                for (i = 0; i< ls.size; i++) {
                    int e = elementAt(&ls, i);
                    push(&L, e);
                }
            }
            printf("\n");
        }
    }


}
```

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

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

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

int topQ(Queue Q) {
    return Q.data[Q.front];
}

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

int emptyQ(Queue Q) {
    return Q.front > Q.rear;
}

void breath_first_search(graph *G) {
    printf("\nDuyet BFS:\n");
    Queue L;
    int mark[100];
    make_null_queue(&L);

    int i, j;
    for(i = 1; i<= G->n; i++)
        mark[i] = 0;

    for( j = 1; j<=G->n; j++) {
        if(mark[j] == 0) {
            pushQ(&L, j);
            printf("Duyet %d\n", j);
            mark[j] = 1;
            while(!emptyQ(L)) {
                int x = topQ(L);
                popQ(&L);
                list ls = neighbors(G, x);
                for ( i = 0; i< ls.size; i++) {
                    int e = elementAt(&ls, i);
                    if(mark[e] == 0) {
                        printf("Duyet %d\n", e);
                        mark[e] = 1;
                        pushQ(&L, e);
                    }

                }
            }
            printf("\n");
        }
    }

}
```

```c
int main(int argc, char const *argv[])
{
    freopen("dothi.txt", "r", stdin);
    int n, m, u, v, i, j;
    scanf("%d%d", &n, &m);
    graph G;
    initGraph(&G, n);
    for(i = 1; i<=m; i++){
        scanf("%d%d", &u, &v);
        addEdge(&G, u, v);
    }
    for(i = 1; i<=n; i++){
        degree(&G, i);
    }
    for(i = 1; i<=n; i++){
        list l = neighbors(&G, i);
        printf("neighbor(%d): ", i);
        for(j = 0; j<l.size; j++){
            printf("%d ", elementAt(&l, j));
        }
        printf("\n");
    }
    depth_first_search(&G);
    breath_first_search(&G);
    return 0;
}
```