```c
#include <stdio.h>

int n =3;

int func1(int arr[][n],int r,int c,int i){
    if(arr[(n+r-1)%n][(n+c-1)%n]>=i){
        return 1;
    }
    else{
        return 0;
    }
}

void func(int arr[][n]){
    int r = 0,c = n/2;
    arr[r][c] = 1;
    for(int i = 2;i<n*n;i++){
        if(func1(arr,r,c,i)){
            r = (n+r-1)%n;
            c = (n+c-1)%n;
        }else{
            r = (r+1)%n;
        }
        arr[r][c] = i;
    }
}

int main(){
    scanf("%d",&n);
    int arr[n][n];
    for(int i = 0; i<n; i++){
        for(int j = 0;j<n;j++){
            arr[i][j] = n*n;}}
    func(arr);
    for(int i = 0; i<n; i++){
        for(int j = 0;j<n;j++){
            printf("%3d ",arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>

void spMatrix(int m,int n,int arr[]){
    int mat[m][n];
    int idx = 0;
    int tp = 0,btm = m-1,lft = 0,rgt = n-1;
    while(tp<=btm&&lft<=rgt){
        for(int i=lft;i<=rgt;i++){
            mat[tp][i] = arr[idx++];
        }tp++;
        for(int i=tp;i<=btm;i++){
            mat[i][rgt] = arr[idx++];
        }rgt--;
        if(tp<=btm){
            for(int i=rgt;i>=lft;i--){
                mat[btm][i] = arr[idx++];
            }btm--;
        }
        if(lft<=rgt){
            for(int i=btm;i>=tp;i--){
                mat[i][lft] = arr[idx++];
            }lft++;
        }
    }
    for(int i=0;i<m;i++){
        for(int j=0;j<n;j++){
            printf("%d\t",mat[i][j]);
        }
        printf("\n");
    }
}

int main(){
    int m,n,arr[100];
    scanf("%d %d",&m,&n);
    for(int i=0;i<m*n;i++){
        scanf("%d",arr+i);
    }
    spMatrix(m,n,arr);
    return 0;
}
```

```c
#include <stdio.h>

int size = 0;
int a = 0, b = 0;
int cnt = 0;
int arr[128][128];

void plop(int x1,int y1, int x2, int y2, int x3, int y3){
    cnt+=1;
    arr[x1][y1] = cnt;
    arr[x2][y2] = cnt;
    arr[x3][y3] = cnt;
}

void solve(int n,int x,int y){
    int r = 0, c = 0;
    if(n==2){
        cnt+=1;
        for(int i = 0;i<n;i++){
            for(int j = 0;j<n;j++){
                if(arr[x+i][y+j]==0){
                    arr[x+i][y+j]=cnt;
                }
            }
        }
        return;
    }
    for(int i = x;i<x+n;i++){
        for(int j = y;j<y+n;j++){
            if(arr[i][j]!=0){
                r = i;c=j;
            }
        }
    }
    if(r<x+n/2&&c<y+n/2){
        plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2-1,x+n/2-1,y+n/2-1);
    }else if(r>=x+n/2&&c<y+n/2){
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2);
        plop(x+n/2-1,y+n/2,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2-1,x+n/2-1,y+n/2-1);
    }else if(r<x+n/2&&c>=y+n/2){
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2-1,x+n/2-1,y+n/2-1);
    }else if(r>=x+n/2&&c>=y+n/2){
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2);
        // plop(x+n/2-1,y+n/2,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        // plop(x+n/2,y+n/2-1,x+n/2,y+n/2,x+n/2-1,y+n/2-1);
        plop(x+n/2-1,y+n/2,x+n/2,y+n/2-1,x+n/2-1,y+n/2-1);
    }
    solve(n/2,x,y);
    solve(n/2,x,y+n/2);
    solve(n/2,x+n/2,y);
    solve(n/2,x+n/2,y+n/2);
}

int main(){
    scanf("%d", &size);
    scanf("%d %d", &a,&b);
    arr[a][b] =-1;
    solve(size,0,0);
    for(int i =0;i<size;i++){
        for(int j=0;j<size;j++){
            printf("%d\t",arr[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <algorithm>
#include <vector>
int count=0;
// int count(std::vector<int>&array){
//     int count=0;
//     for(int i=0;i<array.size();i++){
//         for(int j=i+1;j<array.size();j++){
//             if(array.at(i)<array.at(j)){
//                 count++;
//             }
//         }
//     }
//     return count;
// }

void merge(std::vector<int>&array,std::vector<int>&array1,std::vector<int>&array2){
    array.clear();
    int i,j,k;
    for(i=0,j=0,k=0;i<array1.size()&&j<array2.size();k++){
        if(array1.at(i)<=array2.at(j)){
            array.push_back(array1.at(i++));
            count += array2.size()-j;
        } else if(array1.at(i)>array2.at(j)){
            array.push_back(array2.at(j++));
        }k++;
    }
    while(i<array1.size()){
        array.push_back(array1.at(i++));
    }
    while(j<array2.size()){
        array.push_back(array2.at(j++));
    }
}


void merge_sort(std::vector<int> &array){
    if(1<array.size()){
        std::vector<int> array1(array.begin(),array.begin()+array.size()/2);
        merge_sort(array1);
        std::vector<int> array2(array.begin()+array.size()/2,array.end());
        merge_sort(array2);
        merge(array,array1,array2);
    }
}

int main(){
    int n,val;
    std::cin>>n;
    std::vector<int> array(n);
    array.clear();
    for(int i =0;i<n;i++){
        std::cin>>val;
        array.push_back(val);
    }
    // val = count(array);
    merge_sort(array);
    reverse(array.begin(),array.end());
    for(auto i:array){
        std::cout<<i<<" ";
    }
    std::cout<<std::endl<<count;
    return 0;
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

typedef struct{
    double x,y;
} Point;

typedef struct{
    int u,v;
    double weight;
} Edge;

typedef struct{
    int *parent;
    int *rank;
} UnionFind;

UnionFind* create(int n){
    UnionFind* uf = (UnionFind*) malloc(sizeof(UnionFind));
    uf->parent = (int*)malloc(n*sizeof(int));
    uf->rank = (int*)malloc(n*sizeof(int));
    for(int i=0;i<n;i++){
        uf->parent[i] = i;
        uf->rank[i] = 0;
    }
    return uf;
}

int find(UnionFind* uf,int x){
    if(uf->parent[x]!=x)
        uf->parent[x] = find(uf,uf->parent[x]);
    return uf->parent[x];
}

int unite(UnionFind* uf,int x,int y){
    int rX = find(uf,x);
    int rY = find(uf,y);

    if(rX!=rY){
        if(uf->rank[rX]<uf->rank[rY])
            uf->parent[rX] = rY;
        else if(uf->rank[rX]>uf->rank[rY])
            uf->parent[rY] = rX;
        else{
            uf->parent[rX] = rY;
            uf->rank[rY]++;
        }
    }
}

double calcDistance(Point p1,Point p2){
    return sqrt(pow(p1.x-p2.x,2)+pow(p1.y-p2.y,2));
}

int compareEdges(const void* a,const void* b){
    return ((Edge*)a)->weight > ((Edge*)b)->weight ? 1:-1;
}

double minTotDist(int n,Point* points){
    Edge* edges = (Edge*)malloc(n*n*sizeof(Edge));
    int edgeCount = 0;
    for(int i = 0;i<n;i++){
        for(int j = i+1;j<n;j++){
            double dist = calcDistance(points[i],points[j]);
            edges[edgeCount].u = i;
            edges[edgeCount].v = j;
            edges[edgeCount].weight = dist;
            edgeCount++;
        }
    }
    qsort(edges,edgeCount,sizeof(Edge),compareEdges);

    UnionFind *uf = create(n);
    double totLen = 0;
    for(int i=0;i<edgeCount;i++){
        if(find(uf,edges[i].u)!=find(uf,edges[i].v)){
            unite(uf,edges[i].u,edges[i].v);
            totLen+=edges[i].weight;
        }
    }
    free(uf->parent);
    free(uf->rank);
    free(uf);
    free(edges);
    return totLen;
}

int main(){
    int n;
    scanf("%d",&n);
    Point *points = (Point*) malloc(n*sizeof(Point));
    for(int i=0;i<n;i++){
        scanf("%lf %lf",&points[i].x,&points[i].y);
    }
    double result = minTotDist(n,points);
    printf("%7lf\n",result);
    free(points);
    return 0;
}
```

```cpp
#include <iostream>
#include <vector>
#include <algorithm>

struct Assignment{
    int deadline;
    int time;
    int index;
    bool operator<(const Assignment& other) const{
        if (deadline==other.deadline)
            return time<other.time;
        return deadline<other.deadline;
    }
};


int main(){
    int n;
    std::cin>>n;
    std::vector<Assignment> assignments(n);
    for(int i=0;i<n;i++){
        std::cin>>assignments[i].deadline;
        assignments[i].index = i+1;
    }
    for(int i=0;i<n;i++){
        std::cin>>assignments[i].time;
    }
    sort(assignments.begin(),assignments.end());
    int currentTime = 0,maxLateness = 0;
    for(int i=0;i<n;i++){
        currentTime+=assignments[i].time;
        int late = std::max(0,currentTime-assignments[i].deadline);
        maxLateness = std::max(maxLateness,maxLateness+late);
        std::cout<<assignments[i].index<<" ";
    }
    std::cout<<maxLateness<<std::endl;
    return 0;
}
```

```cpp
#include <iostream>
#include <vector>
#include <string>

int min(int x,int y,int z){
    return std::min(std::min(x,y),z);
}

int edit(std::string x,std::string y){
    int m = x.length(),n = y.length();
    std::vector<std::vector<int>>dp;
    dp.resize(m+1,std::vector<int>(n+1,0));
    for(int i = 0; i<=m; i++){
        for(int j = 0; j<=n; j++){
            if(i==0) dp[i][j]=j*2;
            else if(j==0) dp[i][j]=i*2;
            else if(x[i-1]==y[j-1]) dp[i][j] = dp[i-1][j-1];
            else dp[i][j] = min(dp[i-1][j-1]+1,dp[i-1][j]+2,dp[i][j-1]+2);
        }
    }
    return dp[m][n];
}

int main(){
    std::string x,y;
    std::cin>>x;
    std::cin>>y;
    int ans = edit(x,y);
    std::cout<<ans;
    return 0;
}
```

```cpp
#include <iostream>
#include <vector>
#include <string>
#include <unordered_map>
#include <sstream>

using namespace std;

int minDist(vector<float> &dist , vector <bool> &visited){
    float mindist = 10000,minidx = -1;
    for(int i=0;i<dist.size();i++){
        if(!visited[i]&&dist[i]<mindist){
            mindist = dist[i];
            minidx = i;
        }
    }
    return minidx;
}

void findingpath(vector<vector<float>> &gr,int start,int dest){
    int n = gr[0].size();
    vector<float> dist(n,10000);
    vector<bool> visited(n,false);
    unordered_map<int,int> child;
    dist[start] = 0;
    for(int i = 0; i<n-1 ; i++){
        int u = minDist(dist,visited);
        if(u==-1||u==dest) break;
        visited[u] = true;
        for(int j = 0;j<n;j++){
            if(!visited[j] && gr[u][j] && dist[u]!=10000 && dist[u]+gr[u][j]<dist[j] && gr[u][j]!=1){
                dist[j] = dist[u]+gr[u][j];
                child[u] = j;
            }
        }
    }
    while(start!=dest){
        cout<<start<<endl;
        start = child[start];
    }
    cout<<start<<endl;
}

int main(){
    int n;
    cin>>n;
    vector<vector<float>>gr(n,vector<float>(n,0));
    for(int i=0;i<2*n;i++){
        string x,y;
        cin>>x;
        if(x==".") continue;
        stringstream ss(x);
        while(getline(ss,y,';')){
            string t;
            stringstream ss(y);
            vector <string> v;
            while(getline(ss,t,',')) v.push_back(t);
            if(i<n) gr[i][stoi(v[0])] = stoi(v[1]);
            else gr[i-n][stoi(v[0])] = stoi(v[1])/gr[i-n][stoi(v[0])];
        }
    }
    int start,dest;
    cin>>start>>dest;
    findingpath(gr,start,dest);
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int a[50][50];
int cnt = 0;

void solve(int i, int j, int n, int m, int k){
    if(i==n){
        cnt = (cnt+1)%1000000007;
        return;
    }
    if(j==m){
        solve(i+1,0,n,m,k);
        return;
    }
    int p = -1,q = -1;
    if(j!=0) p = a[i][j-1];
    if(i!=0) q = a[i-1][j];
    for(int x = 1;x<=k;x++){
        if(x!=p && x!=q){
            a[i][j] = x;
            solve(i,j+1,n,m,k);
        }
    }
}

int main(){
    int n,m,k;
    cin>>n>>m>>k;
    solve(0,0,n,m,k);
    cout<<cnt<<endl;
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

double fact[300];
double expVal[300];
int len,maxVal;

int norm(int ind){
    return (ind-(maxVal-1));
}

void calcFact(){
    fact[len] = 1;
    for(int i = len-1;i>=0;--i){
        fact[i] = fact[i+1]*(double(i+1)/(i+maxVal));
    }
}

double individualExp(int occ){
    int sign = 1;
    double retVal = 0;
    for(int l = len,i=0;l>=0;l-=occ,++i){
        double localVal = fact[l];
        for(int j = 1;j<=i;++j)
            localVal*=(double(maxVal - (j-1))/double(j));
        retVal+=(localVal*sign);
        sign*=-1;
    }
    return retVal;
}

int main(){
    cin>>len>>maxVal;
    calcFact();
    double ans = 0;
    expVal[1] = 0;
    for(int occ = 2;occ<=len;++occ)
        expVal[occ] = individualExp(occ);
    for(int occ = 1;occ<len;++occ)
        ans+=(double(occ)*(expVal[occ+1]-expVal[occ]));
    ans+=len*fact[norm(maxVal)];
    cout<<ans<<endl;
    return 0;
}
```

```cpp
#include<cmath>
#include<cstdio>
#include<vector>
#include<iostream>
#include<algorithm>
#include<map>
#include<queue>
using namespace std;

vector<int>a[200005];
int vis[200005]={0},du[200005]={0},dv[200005]={0},vis1[200005]={0};

void mybfsu(int n)
{
    queue<int>p;
    p.push(n);
    du[n]=0;
    while(!p.empty())
    {
        int me=p.front(),i;
        p.pop();
        vis[me]=1;
        for(i=0;i<a[me].size();i++)
        {
            if(vis[a[me][i]]==0)
            {
                du[a[me][i]]=du[me]+1;
                vis[a[me][i]]=1;
                p.push(a[me][i]);
            }
        }
    }
}

void mybfsv(int n)
{
    queue<int>p;
    p.push(n);
    dv[n]=0;
    while(!p.empty())
    {
        int me=p.front(),i;
        p.pop();
        vis1[me]=1;
        for(i=0;i<a[me].size();i++)
        {
            if(vis1[a[me][i]]==0)
            {
                dv[a[me][i]]=dv[me]+1;
                vis1[a[me][i]]=1;
                p.push(a[me][i]);
            }
        }
    }
}
int main()
{
    int n,m,i,temp1,temp2,arr[100],d=0;
    cin>>n>>m;
    for(i=0;i<m;i++)
    {
        cin>>temp1>>temp2;
        a[temp1].push_back(temp2);
        a[temp2].push_back(temp1);
    }
    int u,v,q,c,flag=0;
    cin>>u>>v>>q;

    mybfsu(u);
    mybfsv(v);
    vector<int>queries(q);
    for(i=0;i<q;i++)
    {
        cin>>queries[i];
    }
    for(i=0;i<q;i++)
    {
        if(du[queries[i]]+dv[queries[i]]==du[v])
        {
            cout<<"YES"<<endl;
        }
        else
        {
            cout<<"NO"<<endl;
        }
    }
    return 0;
}
```

```cpp
#include<iostream>
#include<cstdlib>

int DEBUG=0;

int max(int a,int b){
    return a>b ? a:b;
}

int min(int a,int b){
    return a<b ? a:b;
}

int T(int L,int X,int m,int K){
    int res=0;
    if(m==0){
        res=1;
    }else if(X==1){
        if(K>=m && m>L)res=1;
    }else{
        for(int i=0;i<=K;i++){
            if((m-i)>=0){
                res+=T(L-i*X,X-1,m-i,K);
                res%=1000000007;
            }
        }
    }
    return res;
}

int main(int argc,char**argv){
    if(argc>=2){
        DEBUG=atoi(argv[1]);
    }
    int N,L,X,K;
    int m=5;
    if(scanf("%d%d%d%d",&N,&L,&X,&K)==4){
        int sum=0;
        for(int l=L;l<=N;l++){
            sum+=T(l,X,m,K);
            sum%=1000000007;
            if(DEBUG)std::cout<<"l="<<l<<",sum="<<sum<<std::endl;
        }
        std::cout<<sum<<std::endl;
    }
    return 0;
}
```