给定一个字符串，请你将字符串重新编码，将连续的字符替换成“连续出现的个数+字符”。比如字符串 AAAABCCDAA 会被编码成 4A1B2C1D2A。

```
#include <iostream>
#include <string>

using namespace std;

void printString(string str){
    string::size_type i=0;
    int count=1;
for(i=0; i<str.size(); ++i){
        if (str[i] == str[i+1]){
            ++count;
            continue;
        }
        cout << count << str[i];
        count = 1;
    }
}

int main()
{
    string str = "AAAABCCDAA";
    printString(str);

    return 0;
}
```

在一个 N*N 的数组中寻找所有横，竖，左上到右下，右上到左下，四种方向的直线连续 D 个数字的和里面最大的值

卡最后一个案例 的注意题意 ，连续的 D 个数，也就是不包括小于 D 个数的答案， 当然讲道理 D 个数的和
大于 D-1 个数的和，算不算都无所谓，但是不包括有负数的情况，所以会出现极端情况，

D-1 个数的
和大于 D 个数和，所以避免计算小于 D 个数的和，这样不会卡最后一个案例

```
#include<bits/stdc++.h>
using namespace std;
const int N=100;
int arr[N][N];

int main()
{
	int i,j,k;
	int n,d;
	int sum=0;
	int ans[N][N];
	cin>>n>>d;
	for(i=0; i<n; i++){
		for(j=0; j<n; j++){
			cin>>arr[i][j];
		}
	}
	int m1 = 0;//横行
	for(i=0; i<n; i++){
		for(j=0; j<n; j++){
			sum = 0;
			for(k=0; k<d; k++){
				if(j+k>=n) break;
				sum += arr[i][j+k];
			}
			if(m1 < sum){
				m1 = sum;
			}
		}
	}
	//cout<<m1<<endl;
	int m2=0;    //竖列
	for(i=0; i<n; i++){
		for(j=0; j<n; j++){
			sum = 0;
			for(k=0; k<d; k++){
				if(j+k>=n) break;
				sum += arr[j+k][i];
			}
			if(m2 < sum){
				m2 = sum;
```

```
                }
            }
        }
        //cout<<m2<<endl;
        int m3=0;   //左上到右下
        for(i=0; i<n; i++){
            for(j=0; j<n; j++){
                sum = 0;
                for(k=0; k<d; k++){
                    if(i+k>= n ||    j+k >= n) break;
                    sum += arr[i+k][j+k];
                }
                if(m3<sum){
                    m3 = sum;
                }
            }
        }
        //cout<<m3<<endl;
        int m4=0;   //右上到左下
        for(i=0; i<n-d+1; i++){
            for(j=n-1; j>=0; j--){
                sum = 0;
                for(k=0; k<d; k++){
                    if(i+k>=n || j-k<0 ) break;
                    sum += arr[i+k][j-k];
                }
                //cout<<sum<<"    ";
                if(m4<sum){
                    m4 = sum;
                }
            }
        }
        //cout<<m4<<endl;
        int ma=0;
        ma = max(ma,m1);
        ma = max(ma,m2);
        ma = max(ma,m3);
        ma = max(ma,m4);
        cout<<ma<<endl;
        return 0;
}
```

大家一定玩过“推箱子”这个经典的游戏。具体规则就是在一个 N*M 的地图上，有 1 个玩家、1 个箱子、1 个目的地以及若干障碍，其余是空地。玩家可以往上下左右 4 个方向移动，但

是不能移动出地图或者移动到障碍里去。如果往这个方向移动推到了箱子，箱子也会按这个方向移动一格，当然，箱子也不能被推出地图或推到障碍里。当箱子被推到目的地以后，游戏目标达成。现在告诉你游戏开始是初始的地图布局，请你求出玩家最少需要移动多少步才能够将游戏目标达成。

开个四维 vis[x][y][xb][yb]代表人在(x,y)，箱子在(xb,yb)这个状态的最小步数。

```
#include<iostream>
#include<stdio.h>
#include<queue>
using namespace std;
struct q{

        int x,y,xb,yb;

        q(int x,int y,int xb,int yb):x(x),y(y),xb(xb),yb(yb){}

    };

    int a[]={0,0,1,-1},b[]={1,-1,0,0};

    char mp[10][10];

    int vis[10][10][10][10];

    int bx,by,sx,sy,ex,ey,n,m;

    int bfs()

    {

        vis[sx][sy][bx][by]=1;

      q p(sx,sy,bx,by);

        queue<q> que;

        que.push(p);

        while(que.size())

        {

            p=que.front();que.pop();
```

```
            if(p.xb==ex&&p.yb==ey)return vis[p.x][p.y][p.xb][p.yb]-1;
            for(int i=0;i<4;i++)
            {
                int nx=p.x+a[i],ny=p.y+b[i];
                if(nx<0||ny<0||mp[nx][ny]=='#'||nx>=n||ny>=m)continue;
                if(nx==p.xb&&ny==p.yb)
                {
if(nx+a[i]<0||ny+b[i]<0||mp[nx+a[i]][ny+b[i]]=='#'||nx+a[i]>=n||ny+b[i]>=m)continue;
                    if(vis[nx][ny][nx+a[i]][ny+b[i]])continue;
                    vis[nx][ny][nx+a[i]][ny+b[i]]=vis[p.x][p.y][p.xb][p.yb]+1;
                    que.push(q(nx,ny,nx+a[i],ny+b[i]));
                }
                else{
                    if(vis[nx][ny][p.xb][p.yb])continue;
                    vis[nx][ny][p.xb][p.yb]=vis[p.x][p.y][p.xb][p.yb]+1;
                    que.push(q(nx,ny,p.xb,p.yb));
                }
            }
        }
        return -1;
    }
    int main()
```

```
{
    scanf("%d%d",&n,&m);
    for(int i=0;i<n;i++)
        scanf("%s",mp[i]);
    for(int i=0;i<n;i++)
        for(int j=0;j<m;j++)
        if(mp[i][j]=='*')bx=i,by=j;
        else if(mp[i][j]=='X')sx=i,sy=j;
        else if(mp[i][j]=='@')ex=i,ey=j;
    cout<<bfs()<<endl;
    return 0;
}
```

在一条无限长的跑道上，有 N 匹马在不同的位置上出发开始赛马。当开始赛马比赛后，所有的马开始以自己的速度一直匀速前进。每匹马的速度都不一样，且全部是同样的均匀随机分布。在比赛中当某匹马追上了前面的某匹马时，被追上的马就出局。 请问按以上的规则比赛无限长的时间后，赛道上剩余的马匹数量的数学期望是多少

```
import java.util.*;
public class Main{
    static int count = 0;
    public static int remain(int[] array){
        int max = array[0];
        int remain_num = 0;
        for(int i = 0; i < array.length; ++i){
            if(max <= array[i]){
                max = array[i];
                ++remain_num;//这匹马会留下
            }
        }
        return remain_num;
    }
```

```
    public static void swap(int[] array,int i, int j){
        int tmp = array[i];
        array[i] = array[j];
        array[j] = tmp;
    }
    public static void solve(int[] array, int idx){
        if(idx == array.length - 1){
            count += remain(array);
            return ;
        }
        for(int i = idx; i < array.length; ++i){
            swap(array, idx, i);
            solve(array, idx + 1);
            swap(array,idx,i);
        }
    }

    public static void main(String[] arg){
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            int num = sc.nextInt();
            int[] array = new int[num];
            for(int i = 0; i < num; ++i){
                array[i] = i + 1;
            }
            solve(array,0);
            double res = count*1.0;
            for(int j = 0; j < array.length; ++j){
                res = res/array[j];
            }
            String str = String.format("%.4f", res);
            System.out.println(str);
            count = 0;
        }
    }
}
```