A 绝命 DDL

时间限制：1000ms　内存限制：65536kb

通过率：877/1047 (83.76%)　正确率：877/2470 (35.51%)

题目描述

小 w 这个学期的学业特别繁忙，他每周都面临着一波又一波的 ddl。为了高效地应对这些 ddl 以谋求一丝存活的机会，他制定了自己的一套策略来处理它们。小 w 总共面临着 n(0<n<=1000)个 ddl，每个 ddl 有一个完成需要的时间 t 和截止时间 d，小 w 从 0 时刻开始处理这些 ddl，他总是选取**剩余未处理的 ddl 中截止时间最早的一个去做**，如果他无法在这个 ddl 的截止时间前完成它，他将会战略性放弃这个 ddl，去寻找下一个截止时间最早的 ddl。现在小 w 想让你告诉他，**如果从第 0 时刻开始工作**，这 n 个 ddl 他能完成几个？
对于一个完成所需时间为 t_i，截止时间为 d_i 的 ddl，当前时刻为 t，若满足 $t+t_i<=d_i$，则这个 ddl 可以被完成。

输入

第一行一个整数 n(0<n<=1000),
之后 n 行每行两个整数 t_i 和 d_i，分别表示第 i 个 ddl 的完成所需时间和截止时间，两个整数都是 int 范围内的正整数。
保证每个 ddl 的截止时间不相等。

输出

一个整数 count，为可以完成的 ddl 数。

输入样例

```
3
1 2
4 5
3 3
```

输出样例

```
2
```

样例解释

```
第一个选择的 ddl 为"1 2",完成时间为 0 + 1 <= 2,符合要求，
第二个选择的 ddl 为"3 3",完成时间为 1 + 3 > 3,不符合要求,不做这个 ddl，
第三个选择的 ddl 为"4 5",完成时间为 1 + 4 <= 5,符合要求。
因此答案为 2。
```

考察知识点

```
数组递推
难度系数：4
```

题解

这是一道比较基础的题，只要跟着题目描述的步骤实现下来就行。
首先按照截止时间升序将所有的 ddl 排序，之后维护一个当前时间 t，初始值为 0，从头遍历所有的 ddl，对应第 i 个 ddl，若满足 t+t_i <= d_i，则答案计数器加 1，当前时间 t 加 t_i，否则忽略这个 ddl。遍历一遍之后输出结果即可。

参考代码

```
#include<stdio.h>
#include<stdlib.h>
#include<assert.h>
struct ddl{
   int t,d;
} a[1005];
int n,i,t,ans;
int cmp(const void *a,const void *b){
   return (*(struct ddl *)a).d - (*(struct ddl *)b).d;
}
```

```
int main(){
   scanf("%d",&n);
   for (i = 0;i < n;i++)
      scanf("%d%d",&a[i].t,&a[i].d);
   qsort(a,n,sizeof(struct ddl),cmp);
   t = 0;
   ans = 0;
   for (i = 0;i < n;i++)
      if (t + a[i].t <= a[i].d){
         ans++;
         t += a[i].t;
      }
   printf("%d\n",ans);
   return 0;
}
```

B Terry 与神之长子

时间限制：1000ms　内存限制：65536kb

通过率：383/590 (64.92%)　正确率：383/1228 (31.19%)

题目描述

在净化者重新加入达拉姆后，星灵大主教 Artanis 决定为净化者举行一场盛大的阅兵仪式。现在，有一排机械哨兵，Artanis 想请你按照身高从大到小将他们编号。哨兵的数量小于 200000 个。

输入

输入一行，以空格间隔的实数，表示机械哨兵的身高。

输出

按照输入顺序以<编号>:<数值>的格式输出这些数据，要求相同的数值具有相同的编号,各数据之间以空格分隔。数据的数值格式、长度与输入保持不变。

输入样例 1

```
5.3 4.7 3.65 12.345 6e2
```

输出样例 1

```
3:5.3 4:4.7 5:3.65 2:12.345 1:6e2
```

输入样例 2

```
4 5 5 5 5 6
```

输出样例 2

```
3:4 2:5 2:5 2:5 2:5 1:6
```

考察知识点

本题考察了排序与结构体。

难度系数：4

解题思路

本题使用结构体解决思路清晰操作性较好，我们可以用这样一个结构来储存每一个数据：double 类型的身高值，char[]类型的身高值（为了方便按照原格式直接输出），int 类型的 order 值表示输入的顺序，int 类型的 rank 值表示排序后的顺序。这样每个身高值以 char[]类型输入，用 atom 函数转成 double，再将他们按

身高排序后标好 rank 值，最后按照 order 值排序为输入时候的顺序，而后输出。具体操作见代码。

标程

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

struct data_t {
   double value;
   int order;
   int rank;
   char str[30];
}list[200002];

int s_rank(const struct data_t * p1, const struct data_t * p2)
{
if((p1->value - p2->value) < 0) return 1;
else return 0;
}
int s_order(const struct data_t * p1, const struct data_t * p2)
{ return (p1->order - p2->order); }



void gen_rank(struct data_t data[], int n) { //排序后给 rank 值标号
   int i;
   data[0].rank=1;
   for (i=1; i<n;i++)
      if(data[i].value==data[i-1].value)
         data[i].rank=data[i-1].rank;
```

```
      else
         data[i].rank=data[i-1].rank+1;
}



int main() {
   int i,n;
   for (n=0; scanf("%s", list[n].str)!=EOF; n++){
      list[n].value = atof(list[n].str);
      list[n].order = n;
   }


   qsort(list, n, sizeof(struct data_t), s_rank);//先按照大小排序
   gen_rank(list, n);//标 rank
   qsort(list, n, sizeof(struct data_t), s_order);//按照 order 排序为输入的顺序


   for (i=0; i<n; i++) {
      if (i!=0) putchar(' ');
      printf("%d:%s", list[i].rank, list[i].str);
   }
   return 0;
}
```

C 火山喷发

时间限制：1000ms　内存限制：65536kb

通过率：229/390 (58.72%)　正确率：229/1119 (20.46%)

题目描述

做过二维储水问题后考虑下更立体的，这次将水换成岩浆
我们可以把地图划分为 n*m 个格子，每个格子有自己的高度。一个格子的岩浆只能流向上下左右四个方向，并且只能流向高度不大于自己的格子。

现在给你几个火山口的坐标你来看看那些格子有岩浆。

输入

第 1 行有两个整 n,m(1<=n,m<=100)。
之后 n 行每行有 m 个整数描述每个格子的高度 H[i][j](0<=H[i][j]<=10000)
之后 1 行有 1 个整数 kk 表示有 kk 个火山。(1<=k<=n∗m)
之后 k 行每行 2 个整数 x,y 表示火山的位置(1<=x<=n,1<=y<=m)

输出

对于每组数据，输出一个 n∗m 的矩阵，矩阵中 1 表示有岩浆 0 表示没有岩浆。

输入样例

```
3 3

1 1 1

1 2 1

1 1 1

1

1 1
```

输出样例

```
1 1 1

1 0 1

1 1 1
```

考察知识点

二维数组

难度系数：5

题解：

开启两个二维数组，一个(a 存储火山高度，另一个(c 记录火山的喷发情况。

编写函数 hit，作用是实现（x,y）处火山喷发后造成的影响（已经将影响存储进入数组）

检查火山四周的山峰高度，若有较低高度的火山，则蔓延至该区域，同时以该区域作为下一个火山爆发的地点，再次调用 hit 函数，目的是/记录该区域爆发火山造成的影响/

递归开始，注意边界和结束条件，若（x,y）有一个出 n*m 的边界了，就 return 是不错的。

因为会向低处流动，所以流到下一个位置的岩浆也可以作为新的火山口，不会有重复的问题，记得记录递归搜索的历史，注意审题是 k 个火山口同时存在而不是 k 次查找。

代码

```
#include<stdio.h>
int a[110][110];int h[110][110];
int n,m,x,y,k;int addx[]={1,0,-1,0};int addy[]={0,-1,0,1};
int go(int x,int y)
{
   int k;
   if(x<=0||x>m||y<=0||y>n) return 0;
   else if(a[x][y]==1) return 0;
   else
   {
      a[x][y]=1;
      for(k=0;k<4;k++)
```

```
      {
         if(h[x][y]>=h[x+addx[k]][y+addy[k]])
            go(x+addx[k],y+addy[k]);
      }
   }
}
int main()
{
   int i,j;
   while(scanf("%d%d",&n,&m)!=EOF)
   {
      for(i=0;i<=n+1;i++)
         {  a[0][i]=1;a[m+1][i]=1; }
      for(i=0;i<=m+1;i++)
         {  a[i][0]=1;a[i][n+1]=1; }
      for(i=1;i<=n;i++)
         for(j=1;j<=m;j++)
            scanf("%d",&h[j][i]);
      scanf("%d",&k);
      for(i=1;i<=k;i++)
      {
         scanf("%d%d",&y,&x);
         go(x,y);       //递归
      }
      for(i=1;i<=n;i++)      //输出
      {
         for(j=1;j<=m;j++)
         {
            printf("%d ",a[j][i]);
         }
         printf("\n");
```

```
        }
        memset(a,0,sizeof(a));
    }
}
```

D 最后的赢家

时间限制：1000ms　内存限制：65536kb

通过率：503/632 (79.59%)　正确率：503/1133 (44.40%)

题目描述

有 n 个人围成一圈做游戏，1 至 n 编号。从第一个人开始顺序报号 1，2，3。凡报到 3 者退出圈子。最后留在圈子中的人，即为最后的赢家。

编号是一开始的编号，整个过程保持不变

输入

一个大于 1 的正整数，表示有 n 个人

输出

输出最后的赢家的编号

输入样例

```
3
```

输出样例

```
2
```

考察知识点

链表，队列
难度系数：4

解题思路

创建结构体，该结构体是一个链表，保存每个人的编号和一个指针指向下一个人；最后一个人的指针指向第一个人；从头链表开始每隔 2 个链表删除一个链表，直到只剩一个链表指向本身；该链表所对应的编号即是最后的答案。

参考代码

```
#include <stdio.h>
#include <stdlib.h>
#define Len sizeof(struct number)
/* run this program using the console pauser or add your own getch, system("pause") or input loop */


struct number                //构建结构体
{
   int num;                    //用于计数
   struct number*next;
};

int main()
{
   int n;
   struct number*pt;
   struct number *count(int n);
   scanf("%d",&n);
   pt=count(n);                //引用函数
```

```
   printf("%d",pt->num);
   return 0;
}

struct number *count(int n)              //定义返回指针的函数，该指针指向结构体，函数形参为整型
{
   struct number*head,*p1,*p2,*tail;
   int i;
   head=p1=p2=(struct number*)malloc(Len);            //开辟第一个链表
   p2->num=1;                //为第一个链表中的元素赋值
   for(i=2;i<n;i++)
   {
      p1=(struct number*)malloc(Len);                 //不断开辟新链表 ，每个链表指向下一个
      p2->next=p1;
      p1->num=i;
      p2=p1;
   }
   tail=(struct number*)malloc(Len);            //开辟最后一个链表
   p2->next=tail;          //上一个链表指向最后一个链表
   tail->num=n;
   tail->next=head;            //最后一个链表指向头链表

   p1=head;p2=NULL;             //每隔两个链表删除一个链表
```

```
   while(p1!=p2)                                    //最终会形成一个链表指
向自己
     {
       p2=p1->next;
       p1=p2->next;
       p2->next=p1->next;
       p1=p2->next;
     }
   return(p1);              //返回最后剩的链表
}
```

E 末日圈地

时间限制：1000ms　内存限制：65536kb

通过率：55/590 (9.32%)　正确率：55/2040 (2.70%)

题目描述

在一个寒冷的末日环境下，人类依托着巨大的蒸汽枢纽进行取暖，这个枢纽能为周边一定地区提供大量的热量以供生存。在城市规划的过程中，为了枢纽合理分布，需要计算建筑占地面积。现在，给你一个凸多边形的各点坐标，请你求出它的面积

输入

第一个数为点的个数 n(3<=n<=15)

接下来 n 行，每行 2 个整数 x,y 作为点的横纵坐标(0<=x,y<=10000)。

输出

输出一个浮点数，是该多边形的面积。该浮点数保留两位小数。

输入样例

```
4
3 3
3 0
1 0
1 2
```

输出样例

```
5.00
```

考察知识点

```
四则运算
难度系数：5
```

题解

首先出题助教为这道题数据上所出现的问题道歉。原本按照出题本意，所给出点应该是无序的，想考察一下大家使用 qsort 函数进行一次排序，但验数据的时候使用的是对已经排好序的点求面积的代码，因此数据错误，但并未及时发现。这是由于很多同学在题意出错的情况下仍然 AC，经过助教仔细查看代码发现，很多同学用的都是网上他人所写的对已经排好序的点求面积的代码，因此碰巧就 AC 了，希望大家在平常练习的时候多多使用自己所写代码，不要抄袭，也有助于助教发现题目中的问题。

再谈谈这题该如何做，首先需要按照顺时针或者逆时针对点进行排序，完成排序后结合一定平面几何方面的数学知识，进行简单的坐标运算即可。这里详细的数学推导过程不在此阐述，推荐一篇博客 https://blog.csdn.net/lemongirl131/article/details/51130659，其中的阐释比较清晰合理。排序过程其实也是一个难点，可以借助另一个数学概念——叉乘来解决，对于二维向量来说，$A \times B$的结果可看作一个值，当其大于 0 时，B 在 A 的逆时

针方向。相关推导在了解了叉乘的定义后不难理解。

这道题难点在于题目背后的数学几何知识，对这方面数学知识感兴趣的同学还可以深入了解作为一个知识扩充，在充分了解后代码编写难度并不大，这里贴出刘鹄云天同学的代码。

参考代码

```
#include <stdio.h>
int poi[31][2];
int abs(int x)
{
   if(x<0)return -x;
   return x;
}

int cmp(const void *a,const void *b) //借助叉乘，定义比较函数

{
   return   (((int   *)a)[0]-poi[0][0])*(((int   *)b)[1]-poi[0][1])-(((int   *)a)[1]-poi[0][1])*(((int   *)b)[0]-poi[0][0]);
}

int cmp1(const void *a,const void *b) //由于叉乘等于0时，二者共线，因此需要利用纵坐标对那些共线的再次进行排序
{
   return ((int *)a)[1]-((int *)b)[1];
}
int main()
{
    int n,i;
    int  sum=0;
    double ans=0;
    scanf("%d",&n);
    for(i=0;i<n;i++)
       scanf("%d%d",&poi[i][0],&poi[i][1]);
```

```
    qsort(poi,n,sizeof(poi[0]),cmp1);
    qsort(poi[1],n-1,sizeof(poi[0]),cmp);

    for(i=0;i<n-1;i++)   //排序完成后，借助数学公式即可计算得到最
```
终的面积
```
        sum=sum+(poi[i][0]*poi[i+1][1]-
poi[i+1][0]*poi[i][1]);
    sum=sum+(poi[n-1][0]*poi[0][1]-poi[n-1][1]*poi[0][0]);
    ans=0.5*abs(sum);
    printf("%.2f\n",ans);
    return 0;
}
```

F 数组右移.改

时间限制: 1000 ms 内存限制: 65536 kb

总通过人数: 1093 总提交人数: 1171

题目描述

n 个人一起排队接水，第 i 个人需要 a[i]的时间来接水。1≤n≤1000，1≤a[i]≤1000
同时只能有一个人接水，正在接水的人和没有接水的人都需要等待。

完成接水的人会立刻消失，不会继续等待。

你可以决定所有人接水的顺序，并希望最小化所有人等待时间的总和。

输入

第一行一个整数 n
接下来 n 行，每行一个整数表示 a[i]

输出

一行一个整数，表示所有人等待时间的总和的最小值。

输入样例

```
3
1
2
3
```

输出样例

```
10
```

考察知识点：数组综合

解题思路：我们发现接水的时候所有人都需要等待，所以思路便是让时间短的先接水。排序之后不断累加即可。

代码：

```
#include <stdio.h>
int n, a[1020];
void swap(int *a, int *b) {
   int t;
   t = *a; *a = *b; *b = t;
}
void bubble_sort(int x[], int n) {
    int i, j;
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1 - i; j++)
            if(x[j] > x[j + 1])
                swap(x + j, x + j + 1);
```

```
}
int main() {
   int i;
    scanf("%d", &n);
    for (i = 0; i < n; i++) {
       scanf("%d", &a[i]);
    }
    bubble_sort(a, n);
    int s = 0;
    for (i = 0; i < n; i++) {
       s += a[i] * (n - i);
    }
    printf("%d\n", s);
    return 0;
}
```

G 钢管

时间限制：1000ms　内存限制：65536kb

通过率：557/707 (78.78%)　正确率：557/1620 (34.38%)

题目描述

橙橙接到了这样一个任务：从仓库中找出一根钢管。 这听起来不算什么，但是这根钢管的要求可真是让他犯难了，要求如下： 1、 这根钢管一定要是仓库中最长的； 2、 这根钢管一定要是最长的钢管中最细的； 3、 这根钢管一定要是符合前两条的钢管中编码最大的（每根钢管都有一个互不相同的编码，越大表示生产日期越近）。 相关的资料到是有，可是，手工从几百份钢管材料中选出符合要求的那根，这也太难了…… 要不，还是请你编写个程序来帮他解决这个问题吧。

输入

第一行是一个整数 N(1<=N<=10)表示测试数据的组数）
每组测试数据的第一行 有一个整数 m(1<=m<=1000)，表示仓库中所有钢管的

数量，之后 m 行，每行三个整数，分别表示一根钢管的长度（以毫米为单位）、直径（以毫米为单位）和编码（一个 9 位整数）。

输出

对应每组测试数据的输出只有一个 9 位整数，表示选出的那根钢管的编码，每个输出占一行。

输入样例

```
2
2
2000 30 123456789
2000 20 987654321
4
3000 50 872198442
3000 45 752498124
2000 60 765128742
3000 45 652278122
```

输出样例

```
987654321
752498124
```

考察知识点

数组

难度系数：3

解题思路

先定义结构体储存钢管的数据，然后只需要单独定义一个变量记录最符合题意的钢管的编号，然后遍历所有的钢管对它进行更新即可，先比长度，长度短于目前最优的直接淘汰，再比宽度，宽于目前最优的也要淘汰，最后比编码，编码小的也要淘汰，如果更长更细且编码更大，那么更新。参考代码如下：

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

struct wipe
{
int len;
int r;
int num;
};

int main()
{
int t, n, i, max;
struct wipe a[1001];
scanf("%d", &t);
while(t--)
{
    scanf("%d", &n);
    max = 0;
    for(i = 0;i < n; i++)
        scanf("%d%d%d", &a[i].len, &a[i].r, &a[i].num);
    for(i = 0; i < n; i++)
    {
        if((a[i].len > a[max].len)||(a[i].len==a[max].len && a[i].r<a[max].r)||(a[i].len==a[max].len && a[i].r==a[max].r && a[i].num>a[max].num))max = i;
    }
    printf("%d\n",a[max].num);
}
```

```
    }
```

H PPZ 的公共前缀搜索

时间限制：1000ms　内存限制：65536kb

通过率：264/398 (66.33%)　正确率：264/934 (28.27%)

题目描述

没有代码补全功能的 IDE 都是辣鸡！！！PPZ 最近沉迷于 JetBrains 套装不能自拔，其优秀的代码补全功能每天可节省 1000+代码量。那么这么优秀的功能是怎么实现的呢，首先的一点就是要寻找字符串的公共前缀，现在 PPZ 邀请你来牺牲一点头发搞定这个问题

输入

输入共 n+1 行，第一行是一个整数 n(0<n<=10)， 接下来的 n 行，每行一个字符串，只由大写字母和小写字母组成。

输出

输出为一行，输出所有输入字符串的公共前缀部分，若没有公共前缀输出 None 例如要判断的字符串为”abv","abbbbbbbb","abcde"，输出为“ab"； 注：本题目不区分大小写字母，最后输出一律转换为小写，也就是说，要判断的字符串为”Abv", "aBbbbbbbb", "abcde" 时，输出为“ab"。

输入样例

```
3
field
fIll
```

```
FIbonacci
```

输出样例

```
fi
```

考察知识点

```
字符串操作
难度系数：6
```

解题思路

根据题目要求设定好数据范围，将读入的第一个字符串作为基准字符串，再根据 n 的值依次读入字符串，先取二者长度较小的值，然后从字符串开头进行字符匹配，使用 tolower 函数转换为小写。最后补齐字符串结束标志'\0'，并根据 min 的值进行输出（n 为 1 时输出原字符串）。

例程

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define N (20+1)
int main()
{
    int n,i,min;
    scanf("%d",&n);
    char result[N],input[N];
    scanf("%s",result);
    min=strlen(result);
```

```
    for(i=1;i<n;++i){
        scanf("%s",input);
        if(strlen(input)<min)
            min=strlen(input);
        int j=0;
        while(j<min&&tolower(result[j])==tolower(input[j]))
            j++;
        min=j;
    }
    result[min]='\0';
    for(i=0;i<min;++i)
        result[i]=tolower(result[i]);
    printf("%s\n",min==0?"None":result);
    return 0;
}
```

I 寻找爱好相同的人

时间限制：1000ms　内存限制：65536kb

通过率：474/606 (78.22%)　正确率：474/1257 (37.71%)

题目描述

每个人有三项爱好,分别是食物，饮料，电影，运动等中的任意三项,第 i 个人的三种爱好分别用一个整数 ai,bi,ci 来表示. 现在给出 n 个人的爱好,如果两个人起码有两项以上的爱好对应的数字相同,那么我们认为这两个人具有相同的爱好,请问一共有多少对人拥有相同的爱好. 满足 $a_i==a_j,b_i==b_j,c_i==c_j$ 分别算作一种爱好相同。

输入

第一行,一个整数 n(1<=n<=100)
接下来的 n 行,每行三个整数 ai,bi,ci,用空格分割(0<=ai,bi,ci<=10)

输出

一个整数,表示拥有相同爱好的人的对数，
举例:假如 1,2,3 人都有相同的爱好,那么有(1,2) (1,3) (2,3) 3 对人有相同的爱好

输入样例

```
4
1 2 3
1 2 4
1 2 3
2 2 3
```

输出样例

```
5
```

考察知识点

```
四则运算
难度系数：4
```

思路解析:

题目比较简单,可以使用结构体(让同学们熟练结构体的使用)或者三个一维数组,然后二维循环判断每两个人之间的关系,因为两个人的关系只需要判断一遍,所以不需要和自己之前的人比,j 从 i+1 开始.然后比较 i,j 两人是否爱好相同.

题目代码:

```
#include<stdio.h>
#define MAXN 100
struct node{
   int a,b,c;
}p[MAXN +10];
```

```
int main()
{
   int n,i,j,ans = 0;
   scanf("%d",&n);
   for(i = 1;i <= n;i++)
      scanf("%d%d%d",&p[i].a,&p[i].b,&p[i].c);
   for(i = 1;i <= n;i++)
      for(j = i + 1;j <= n;j++)
         if((p[i].a == p[j].a) + (p[i].b == p[j].b) + (p[i].c == p[j].c) >= 2)
            ans++;
   printf("%d\n",ans);
   return 0;
}
```

J 多边形判断的问题

时间限制：1000ms　内存限制：65536kb

通过率：740/809 (91.47%)　正确率：740/1370 (54.01%)

题目描述

有一根长长木条，随机选 k 个位置把它切成(k+1)(段短木条。

求这些短木条能组成一个多边形的概率 p。

（提示：当其中一个短木条大于等于长木条的距离一半时，这些短木条无法组成多边形。）

输入

整数 k（30 以内的正整数）。

输出

整数 k 所对应的概率值 p。　（结果保留 5 位小数，要求四舍五入。）

输入样例

```
3
```

输出样例

```
0.50000
```

样例解释

不难发现本题的答案与木条长度无关。
在一条直线上切，似乎难以处理，可以把直线接成一个圆，多切一下，即在圆上随机选 k+1 个点，把圆周切成 k+1 段。根据对称性，两个问题的答案相同。
新问题就容易处理得多了，正难则反，“组不成多边形”的概率就是其中一个小木条至少跨越了半个圆周的概率。设这个最长的小木条从点 i 开始逆时针跨越了至少半个圆周，则其他所有点都在这半个圆周之外。
除了点 i 之外其他每个点位于这半个圆周以外的概率均为 1/2，因此概率为 1/2^k。
因为第一个切点(将圆切成线段的那一刀)可以在(k+1)个点中任选，所以“组不成多边形”的概率就是(k+1)*(1/2)^k，可以组成多边形的概率就是用 1 减去组不成多边形的概率，样例中 k=3 时，(k+1)*(1/2)^k=0.5，1-0.5=0.5，最后结果是 0.5。

考察知识点

四则运算

难度系数：9

考察知识点

本题实际上考察的是正难则反的思维方式，具体的公式推导已经在样例解释中给出，只需计算 1-*(k+1)*(1/2)^k 并保留 5 位小数即可，参考程序如下：

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

```
int main()
{
    int k,i;
    double p;
    scanf("%d",&k);
    p=(double)(k+1);
    for(i=1;i<=k;i++) p=p/2 ;
    printf("%.5f",1-p);
    return 0;
 }
```