第二十三届全国青少年信息学奥林匹克联赛初赛提高组解析

C++语言试题

竞赛时间：2017 年 10 月 14 日 14:30~16:30

选手注意：

- 试题纸共有 10 页，答题纸共有 2 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

一、单项选择题（共 **15** 题，每题 **1.5** 分，共计 **22.5** 分；每题有且仅有一个正确选项）

1.从（ ）年开始，NOIP 竞赛将不再支持 Pascal 语言。

A. 2020　　B. 2021　　C. 2022　　D. 2023

解析：

CCF 关于 NOI 系列赛事程序设计语言变更的公告

根据国际信息学奥林匹克竞赛（IOI）的相关决议并考虑到我国目前程序设计语言的具体情况，CCF 决定：

1. 2020 年开始，除 NOIP 以外的 NOI 系列其他赛事（包括冬令营、CTSC、APIO、NOI）将不再支持 Pascal 语言和 C 语言；

2. 从 2022 年开始，NOIP 竞赛也将不再支持 Pascal 语言。即从 NOIP2022 开始，NOI 系列的所有赛事将全部取消 Pascal 语言。

在无新增程序设计语言的情况下，NOI 系列赛事自 NOIP2022 开始将仅支持 C++语言。

此通知。

中国计算机学会 2016 年 11 月 1 日

2. 在 8 位二进制补码中，10101011 表示的数是十进制下的（ ）。

A. 43　　B. -85　　C. -43　　D. -84

解析：

在计算机内部，不论正整数和负整数，均以补码形态存储。负数补码形态稍复杂，正数则简单：$[正数]_{原码}=[正数]_{反码}=[正数]_{补码}$，因此，我们可以从正数入手，将正数与其对应的负值相加结果为 0，而 0 在计算机内部的存储为 0，从这一点反推出负数补码。

	10101011	
+	01010101	$(01010101)_2=1*2^6+1*2^4+1*2^2+1*2^0=(85)_{10}$
	00000000	

因此，答案为-85

3. 分辨率为 1600x900、16 位色的位图，存储图像信息所需的空间为（ ）。

A. 2812.5KB　　B. 4218.75KB　　C. 4320KB　　D. 2880KB

解析：

高中信息技术学考内容，位图（bmp 格式）存储容量的计算公式：

存储容量＝水平像素数*垂直像素数*位数/8/1024 （单位：KB）

=1600*900*16/8/1024=2812.5KB，千万别除 1000 进行估算，否则容易选 C，答案选 A

4. 2017 年 10 月 1 日是星期日，1949 年 10 月 1 日是（ ）。

A. 星期三　　B. 星期日　　C. 星期六　　D. 星期二

解析：

先确定边界，例如：从 2017 年 1 月 1 日至 2017 年 12 月 31 日（天数+364），则算到 2018 年 1 月 1 日（天数+365）

① 因此，统计 1949 年——2017 年，完整的年份（2017-1949=68 年），则需要累加天数（+68*365=24820）

② 瑞年的统计 1952 年——2016 年（=（2016-1952）/4+1=17）

③ (24820+17)%7=1

因此，1949 年 10 月 2 日刚好与 2017 年 10 月 1 日一样都是星期日，1949 年 10 月 1 日则是星期六，答案选 C

5. 设 G 是有 n 个结点、m 条边（n ≤ m）的连通图，必须删去 G 的（ ）条边，才能使得 G 变成一棵树。

A. m – n + 1　　B. m - n　　C. m + n + 1　　D. n – m + 1

解析：

首先，了解一棵树，当你为这棵树画好 1 个根节点后，你再添加 1 个节点同时，你都会添加与之对应的 1 条边。所以，任何一棵树，均有：结点数量和=边数和+1

所以这棵树的边数：n-1，需要从图 G 中删除的边数为：m-(n-1)=m-n+1，答案选 A

6. 若某算法的计算时间表示为递推关系式：

T(N) = 2T(N / 2) + N log N

T(1) = 1

则该算法的时间复杂度为（ ）。

A. O(N)　　B. O(N log N)　　C. O(N $\log^2$ N)　　D. O(N^2)

解析：

因为是选择题，所以我选择直接代入数值，然后对比，进行估算。

T(k)	计算	O(N)
T(1)	1	1
T(2)	=2T(1) + 2 log 2=2*1+2*1=4	4
T(4)	=2T(2) + 4 log 4=2*4+4*2=16	16
T(8)	=2T(4) + 8 log 8=2*16+8*3=56	56
T(16)	=2T(8) + 16 log 16=2*56+16*4=176	176
T(32)	=2T(16) + 32 log 32=2*176+32*5=512	512
……	……	……
T(N)		O(N $\log^2$ N)

选取较大的 N 值，对各选项进行评估，答案选 C

7. 表达式 a * (b + c) * d 的后缀形式是（ ）。

A.a b c d * + *　　B. a b c + * d *　　C. a * b c + * d　　D. b + c * a * d

解析：

中缀表达式指的是运算符在中间，操作数在两端，比如：a+b，后缀表达式运算符在后面，如：a b +，后缀表达式不需要括号，在程序设计中通常借助栈来完成运算。

a * (b + c) * d = a * (b c +) * d = (a b c + *) * d=a b c + * d *

因此，答案选 B

8. 由四个不同的点构成的简单无向连通图的个数是（ ）。

A. 32　　B. 35　　C. 38　　D. 41

解析：

4 个点的无向连通图，边数总共=3+2+1=6，给每一条边标记编号，我们可以从边数入手，分类来讨论图的构造：

图中边的数量	图的种类	说明
6 条边	1	完全图
5 条边	C（6,1）=6	选其中 1 条边删除
4 条边	C（6,2）=6*5/(2*1)=15	选其中 2 条边删除
3 条边分析：	C（6,3）=6*5*4/(3*2*1)=20	★需排除不是连通图的情况
存在孤点(另 3 个点彼此连通)	C（4,1）=4	孤点选择方案
3 条边有效方案数：	=20-4=16	
1、2 条边	0	不再是连通图了
总方案：1+6+15+16=38		

9. 将 7 个名额分给 4 个不同的班级，允许有的班级没有名额，有（ ）种不同的分配案。

A. 60　　B. 84　　C. 96　　D. 120

解析：

对 4 个不同班级分配结果进行分类讨论：

分配方案	计算方式	结果
7 个名额分到 1 个班	C（4,1）=4	4
7 个名额在到 2 个班	（1、6）=A（4,2）=4*3	12*3=36
	（2、5）=A（4,2）=4*3	
	（3、4）=A（4,2）=4*3	
7 个名额在到 3 个班	（1、1、5）=A（4,3）/A（2,2）=12	12*3+24=60
	（1、3、3）=A（4,3）/A（2,2）=12	
	（2、2、3）=A（4,3）/A（2,2）=12	
	（1、2、4）=A（4,3）=4*3*2=24	
7 个名额在到 4 个班	（1、1、1、4）=C（4,1）=4	4*2+12=20
	（1、1、2、3）=C（4,1）*A（2,2）=12	
	（1、2、2、2）=C（4,1）=4	
总方案：4+36+60+20=120		

10. 若 f[0] = 0, f[1] = 1, f[n + 1] = (f[n] + f[n - 1]) / 2，则随着 i 的增大，f[i]将接近于（ ）。

A. 1/2　　B. 2/3　　C. $\frac{\sqrt{5}-1}{2}$　　D. 1

解析：

（方法 1：）画数轴标点表示，比较直观。

（方法 2：）因为，f(n+1)=[f(n)+f(n−1)]/2

所以有：2f(n+1)−2f(n)=f(n−1)−f(n)

f(n+1)−f(n)=−1/2*[f(n)−f(n−1)]

设 g(n)=f(n+1)−f(n)

g(n)=−1/2*g(n−1)

g(0)=f(1)−f(0)=1

$g(n)=(-1/2)^n$

$f(n)=[f(n)-f(n-1)+f(n-1)-f(n-2)+\cdots\cdots+f(1)-f(0)]+f(0)$

$=g(n-1)+g(n-2)+\cdots\cdots+g(0)+0$

$=(-1/2)^{n-1}+(-1/2)^{n-2}+\cdots\cdots+(-1/2)^0$

$=2/3[1-(-1/2)^n]$

$=2/3$

等比数列求和：（1−q）*Sn=a1−a1*q^n

所以，当公比不为 1 时，等比数列的求和公式为 Sn=[a1*(1−q^n)]/(1−q)

对于一个无穷递降数列，数列的公比小于 1，当上式得 n 趋向于正无穷大时，分子括号中的值趋近于 1，取极限即得无穷递减数列求和公式

S=a1/(1−q)

$$= \text{Limit}\left[\frac{2}{3}\left(1-\left(-\frac{1}{2}\right)^n\right), n \to \infty\right]$$

$$= \text{Limit}\left[\frac{2}{3}(1-2^{-n}(-1)^n), n \to \infty\right]$$

$$= \frac{2}{3}$$

“极限”是数学中的分支——微积分的基础概念，广义的“极限”是指“无限靠近而永远不能到达”的意思。

数学中的“极限”指：某一个函数中的某一个变量，此变量在变大（或者变小）的永远变化的过程中，逐渐向某一个确定的数值 A 不断地逼近而“永远不能够重合到 A”（“永远不能够等于 A，但是取等于 A”已经足够取得高精度计算结果）的过程中，此变量的变化，被人为规定为“永远靠近而不停止”、其有一个“不断地极为靠近 A 点的趋势”。

极限是一种“变化状态”的描述。此变量永远趋近的值 A 叫做“极限值”（当然也可以用其他符号表示）。

11. 设 A 和 B 是两个长为 n 的有序数组，现在需要将 A 和 B 合并成一个排好序的数组，请问任何以元素比较作为基本运算的归并算法最坏情况下至少要做（　　）次比较。

A. n^2　　B. n log n　　C. 2n　　D. 2n-1

解析：

数组 A：

1	2	3	……	n−1	n
2	4	6	……	10	14

数组 B：

1	2	3	……	n−1	n
1	3	5	……	11	13

上面图表中给出的仅是其中的一个特例，是其中最坏的一种情况。由表中可见，每次比较都是从数组 A 或数组 B 中选出一个元素。直到其中的一个数组恰好剩余 1 个元素，而另一个数组中的元素刚好取光时结束。

所以，比较的总次数是 2*n−1，答案选 D

12. 在 n（n ≥ 3）枚硬币中有一枚质量不合格的硬币（质量过轻或质量过重），如果只有一架天平可以用来称重且称重的硬币数没有限制，下面是找出这枚不合格的硬币的算法。请把 a-c 三行代码补全到算法中。

a. $A \leftarrow X \cup Y$

b. $A \leftarrow Z$

c. $n \leftarrow |A|$

```
算法 Coin(A, n)
1.  k ← ⌊n/3⌋
2.  将 A 中硬币分成 X，Y，Z 三个集合，使得|X| = |Y| = k, |Z| = n − 2k
3.  if W(X) ≠ W(Y)      //W(X), W(Y)分别为 X 或 Y 的重量
4.  then _____  // 缩小问题规模范围，不合格硬币只存在：XUY中
5.  else _____  // 不合格硬币只存在：Z中
6.  _____       // 更新当前不合格硬币所在的集合元素个数：n = |A|
7.  if n>2 then goto 1
8.  if n=2 then 任取 A 中 1 枚硬币与拿走硬币比较，若不等，则它不合格；若相等，
    则 A 中剩下的硬币不合格.
9.  if n=1 then A 中硬币不合格
```

正确的填空顺序是（ ）。

A. b, c, a　　B. c, b, a　　C. c, a, b　　D. a, b, c

解析（简单题，技术学考题）：

运用三分法，利用天平进行称重。将硬币大致分成三等份，其中|X| = |Y|=k, |Z| = $n - 2 * k$

13. 有正实数构成的数字三角形排列形式如图所示。第一行的数为 a_{11}；第二行的数从左到右依次为 a_{21}, a_{22}；…第 n 行的数为 $a_{n1}, a_{n2}, \cdots, a_{nn}$。从 a_{11} 开始，每一行的数 a_{ij} 只有两

条边可以分别通向下一行的两个数 $a_{(i+1)j}$ 和 $a_{(i+1)(j+1)}$。用动态规划算法找出一条从 a_{11} 向下通到 $a_{n1}, a_{n2}, \cdots, a_{nn}$ 中某个数的路径，使得该路径上的数之和达到最大。

令 C[i,j]是从 a_{11} 到 a_{ij} 的路径上的数的最大和，并且 C[i,0]=C[0,j]=0，则 C[i,j]=（ ）。

max{C[i-1,j-1], C[i-1,j]} + a_{ij}

C[i-1,j-1] + C[i-1,j]

max{C[i-1,j-1], C[i-1,j]} + 1

max{C[i,j-1],C[i-1,j]} + a_{ij}

解析（简单题，动规入门题）：

将数字三角形整理、数形转换，将其转换成数组形态，进行观察：

a_{11}							
a_{21}	a_{22}						
a_{31}	a_{32}	a_{33}					
a_{41}	a_{42}	A_{43}	a_{44}				
……	……						
a_{n1}	a_{n2}	……					a_{nn}

由图可见，元素 c[i, j]是从 a[i-1, j-1]或 a[i-1, j]中选择其中的较大值：

$$c[i,j] = \max\{c[i-1,j-1], c[i-1,j]\} + a[i,j]$$

答案选 A

14. 小明要去南美洲旅游，一共乘坐三趟航班才能到达目的地，其中第 1 个航班准点的概率是 0.9，第 2 个航班准点的概率为 0.8，第 3 个航班准点的概率为 0.9。如果存在第 i 个（i=1,2）航班晚点，第 i+1 个航班准点，则小明将赶不上第 i+1 个航班，旅行失败；除了这种情况，其他情况下旅行都能成功。请问小明此次旅行成功的概率是（ ）。

A. 0.5　　B. 0.648　　C. 0.72　　D. 0.74

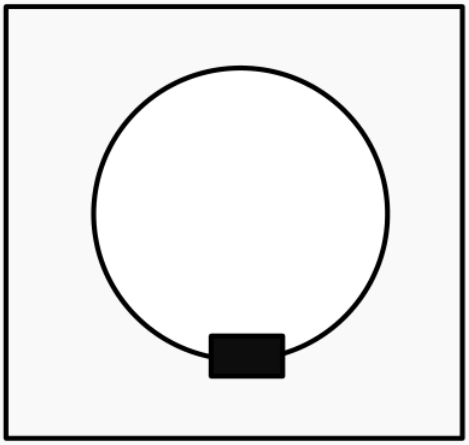
解析（简单题，数学排列组合题）：

根据题意，列举旅行不成功情形，分类探讨：

旅行不成功情况	计算
第 1 航晚，第 2 航准	(1-0.9)*0.8=0.08
第 1 航晚，第 2 航晚，第 3 航准	(1-0.9)*(1-0.8)*0.9=0.018
第 1 航准，第 2 航晚，第 3 航准	0.9*(1-0.8)*0.9=0.162
不成功率总计：	0.08+0.018+0.162=0.26
成功率：	1-0.26=0.74

答案选 D

15. 欢乐喷球：儿童游乐场有个游戏叫“欢乐喷球”，正方形场地中心能不断喷出彩色乒乓球，以场地中心为圆心还有一个圆形轨道，轨道上有一列小火车在匀速运动，火车有六节车厢。假设乒乓球等概率落到正方形场地的每个地点，包括火车车厢。小朋友玩这个游戏时，只能坐在同一个火车车厢里，可以在自己的车厢里捡落在该车厢内的所有乒乓球，每个人每次游戏有三分钟时间，则一个小朋友独自玩一次游戏期望可以得到（　　）个乒乓球。假设乒乓球喷出的速度为 2 个/秒，每节车厢的面积是整个场地面积的 1/20。



A. 60　　B. 108　　C. 18　　D. 20

解析（简单题，小学数学题）：

由于乒乓球是等概率落到正方形场地每个点上，因此火车车厢接球数量与车厢面积成等比例关系。每秒发 2 个，3 分钟内，总共发出的小球数量：3*60*2=360 个。

由于一节车厢的面积是整个场地面积的 1/20，因此一节车厢期望得到的小球数量是：360*（1/20）=18 个

答案选 C

二、不定项选择题（共 **5** 题，每题 **1.5** 分，共计 **7.5** 分；每题有一个或多个正确选项，多选或少选均不得分）

1. 以下排序算法在最坏情况下，时间复杂度最优的有（　）。

A. 冒泡排序　　B. 快速排序　　C. 归并排序　　D. 堆排序

解析（简单题，数据结构常识题）：

一般来讲，算法时间复杂度只考虑平均情况和最坏情况两种情形，在大部分情况下，算法的平均情况和最坏情况基本等同，具有研究价值。

但也存在例外，比如快速排序，在排序过程中，如果每次选择参与比较的 key 都是当前区域的第 1 元素，而待排序的区域恰好是有序的，这种情况下的数据划分将变得非常糟糕，时间复杂度降为 $O(N^2)$。归并排序和堆排序不存在这种情况，为：O（NlogN），答案选 CD

2. 对于入栈顺序为 a, b, c, d, e, f, g 的序列，下列（　）不可能是合法的出栈序列。

A. a, b, c, d, e, f, g　　B. a, d, c, b, e, g, f

C. a, d, b, c, g, f, e　　D. g, f, e, d, c, b, a

解析（简单题，数据结构常识题）：

由于入栈是按照字母的先后顺序进入的，因此出栈的顺序必然是当前字母与其后方的（小于当前字母）字母逆序构成，才是有效的。

a，d，b，c，g，f，e

（字母 d 和后方小于 d 的字母，应呈现出的顺序是 d c b，字母间隔位置可以不连续）

答案选 C

3. 下列算法中，（ ）是稳定的排序算法。

A. 快速排序　　B. 堆排序　　C. 希尔排序　　D. 插入排序

解析（简单题，数据结构常识题）：

排序的稳定性指的是 a1、a2 这两个元素（若 a1=a2），若排序开始前 a1 位于 a2 前面，则在整个排序过程中，自始至终确保 a1 均在 a2 前面，则该排序是稳定的。

算法是从快速排序左侧找一元素，右侧也找一个元素，然后交换，如果中间有元素和交换元素相同，则排序就不稳定。

堆排序，当堆构建成功后，我们将堆顶元素和数组最后一个元素交换，如果数组中存在元素与数组最后位置元素相同，将其交换到堆顶，该操作将直接导致堆排序的不稳定。

希尔排序，是等间距插入排序或冒泡排序，如果当前待处理的都是等间距序列元素，在插入元素 A 过程中，如果元素 A 移动的时候，中间跨段数据序列中有其它元素和元素 A 相同，则出现排序的不稳定。

答案选 D

4. 以下是面向对象的高级语言的有（ ）。

A. 汇编语言　　B. C++　　C. Fortran　　D. Java

解析（简单题，计算机常识题）：

汇编语言面向机器底层，是低级语言，执行效率极高，Fortran 语言主要用于工程和科学计算，是结构化的编程语言。

答案选 BD

5. 以下和计算机领域密切相关的奖项有（ ）。

A. 奥斯卡奖　　B. 图灵奖　　C. 诺贝尔奖　　D. 王选奖

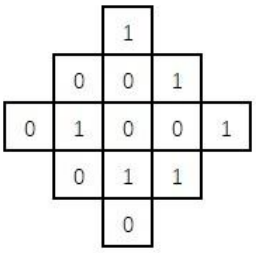
解析（王选奖推广题）：

中国计算机学会王选奖(简称“CCF 王选奖”)原名“中国计算机学会创新奖”(简称“CCF 创新奖”)，于 2005 年创立，每年评选一次，属于社会力量设立的科学技术奖。该奖旨在推动中国计算机及相关领域的科技创新和进步，促进科研成果的转化和 IT 产业的发展，推动科技界学术共同体评价体系的建立，发现和激励创新型科技人才。2006 年，为了纪念著名科学家王选院士为中国计算机事业做出的非凡贡献，学习他严谨、务实、奉献、创新、勇于超越的科研精神，中国计算机学会决定将中国计算机学会创新奖以王选院士的名字命名，更名为中国计算机学会王选奖。

答案选 BD

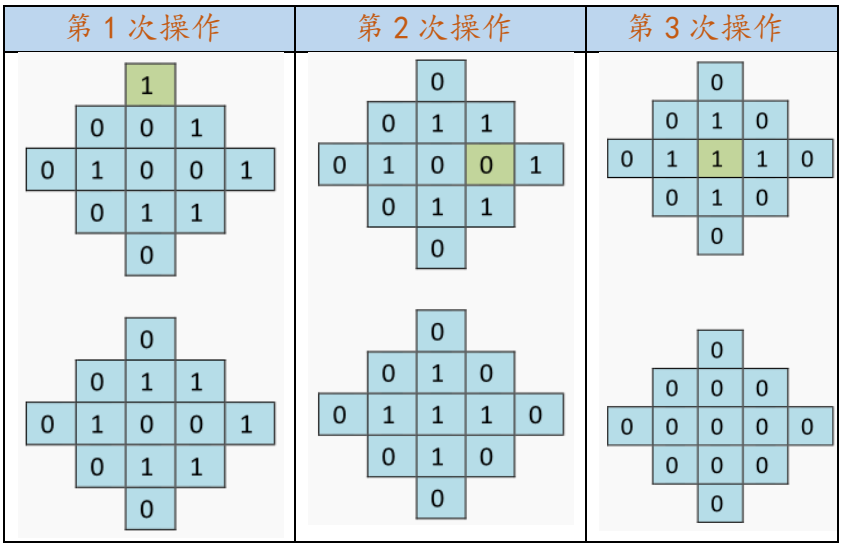
三、问题求解（共 **2** 题，每题 **5** 分，共计 **10** 分）

1. 如右图所示，共有 13 个格子。对任何一个格子进行一次操作，会使得它自己以及与它上下左右相邻的格子中的数字改变（由 1 变 0，或由 0 变 1）。现在要使得所有的格子中的数字都变为 0，至少需要_____次操作。



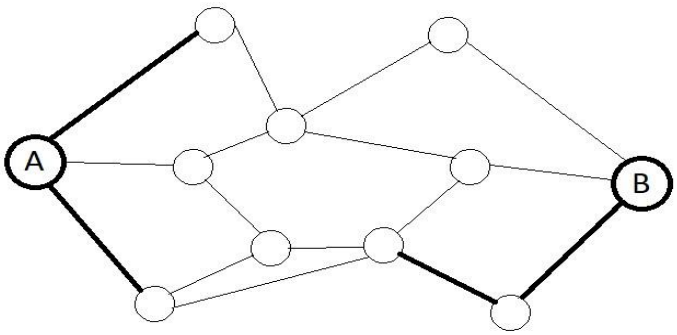
解析（方法 1 观察）：

具体操作如下所示：



答案 3

2. 如下图所示，A 到 B 是连通的。假设删除一条细的边的代价是 1，删除一条粗的边的代价是 2，要让 A、B 不连通，最小代价是_____（2 分），最小代价的不同方案数是_____（3 分）。（只要有一条删除的边不同，就是不同的方案）



解析（方法 1 观察）：

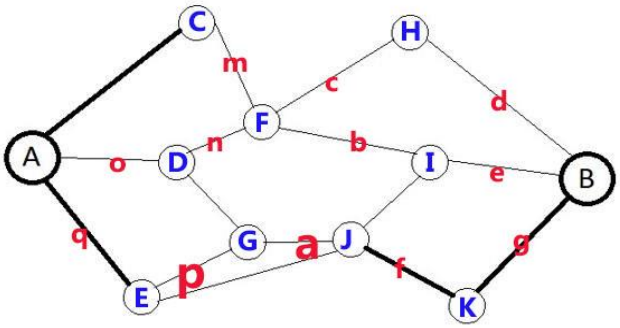
2.1 答案 4

要让 A 和 B 不连通，方案存在多种，其中最小代价是 4

首先，给图中对应的结点及边做编号。

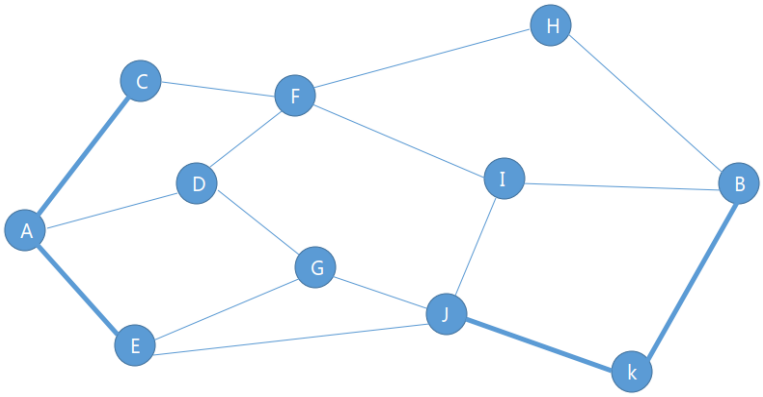
2.2 答案 9，可行方案罗列如下：

a-b-c	f-e-d	a-n-m
a-b-d	g-e-c	p-o-m
f-e-c	g-e-d	q-o-m



方法二、理论指导：

（1）将无向图结点做编号，如下：



（2）知识点回顾：平面图性质

1. （欧拉公式）如果一个连通的平面图有 n 个点，m 条边和 f 个面，那么 f=m-n+2
2. 每个平面图 G 都有一个与其对偶的平面图 G*

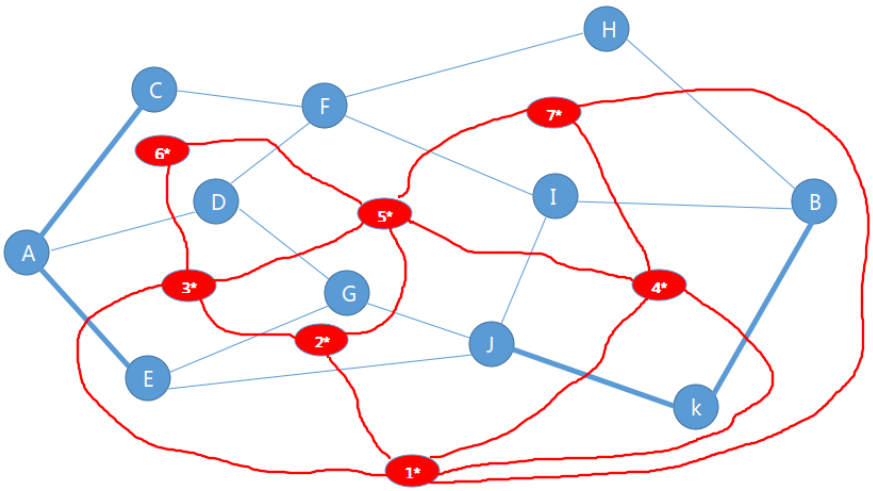
   G*中的每个点对应 G 中的一个面（这里我用数字*来标记图中的面）

   对于 G 中的每条边 e

   e 属于两个面 f1、f2，加入边(f1*， f2*)

   e 只属于一个面 f，加入回边(f*， f*)

接下来，上图的**对偶图 G***如下所示（为了漂亮，我没有把图 G*画完整）：



（3）知识点回顾：平面图与其对偶图的关系

- 平面图 G 与其对偶图 G*之间存在怎样的关系呢？
  G 的面数等于 G*的点数，G*的点数等于 G 的面数，G 与 G*边数相同
  G*中的环对应 G 中的割一一对应

s-t 平面图上最大流的快速求法

如何利用这些性质？

直接求解

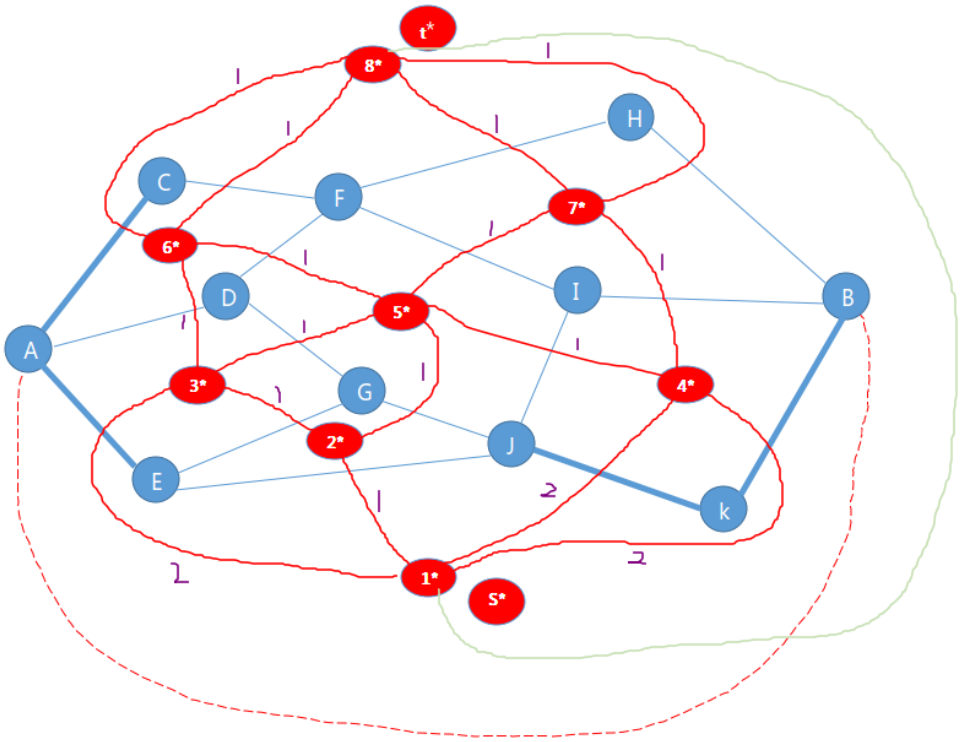
仍然困难

利用最大流—最小割定理转化模型

根据平面图与其对偶图的关系，想办法求出最小割

（4）知识点回顾：利用最短路求最小割

- 对于一个 s-t 平面图，我们对其进行如下改造：
  1. 连接 s 和 t，得到一个附加面
  2. 求该图的对偶图 G*，令附加面对应的点为 s*，无界面对应的点为 t*
  3. 删去 s*和 t*之间的边



这里，你只需要看红色结点及红色的实心线条，红色实线每一条边对应的值与蓝色交叉，则边权值与蓝线的权值一样，图中，我用紫色数字标记出来了。

利用最短路求最小割

一条从 s*到 t*的路径，就对应了一个 s-t 割！

更进一步，如果我们令每条边的长度等于它的容量，那么最小割的容量就等于最短路的长度！

- 分析一下时间复杂度

  新图中的点数和边数都是 O(n)的

  使用二叉堆优化的 Dijkstra 算法求最短路，时间复杂度为 O(nlog2n)

接下来，我用最短路方法来求最小割：

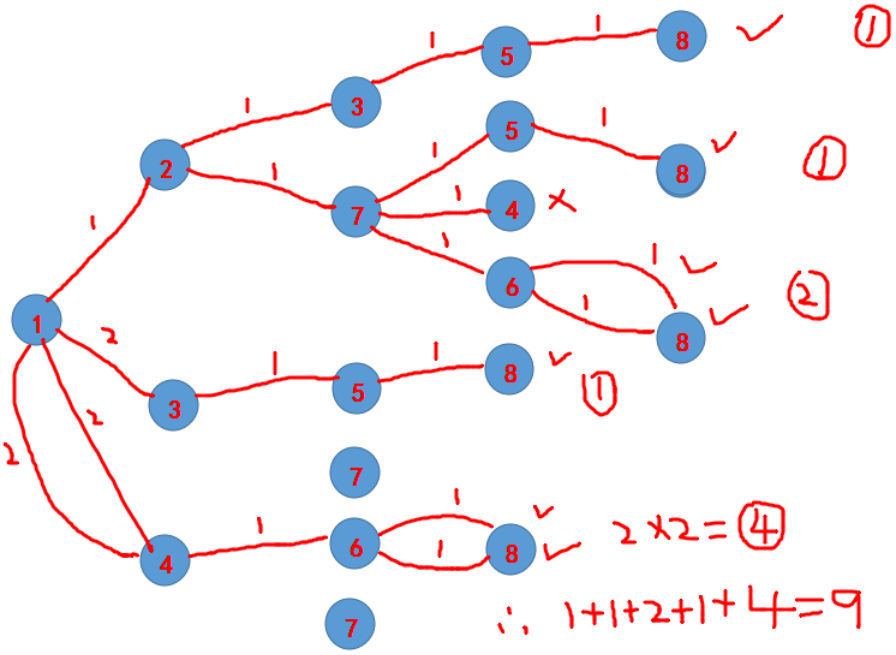
同时，我将每一个已经求出最短路的结点权值保存在数组 dis[ ]中：

1	2	3	4	5	6	7	8
0	1	2	2	3	3	2	4

Dijkstra 算法求最短路是基础算法，我就不介绍了吧，直接填上表。

因此，该图最小代价（最小割）答案是 4

接下来，根据最小割值，结合上图，我利用 bfs 思想找不同方案数，直接给图吧：



四、阅读程序写结果（共 **4** 题，每题 **8** 分，共计 **32** 分）

1. `#include <iostream> using namespace std;`

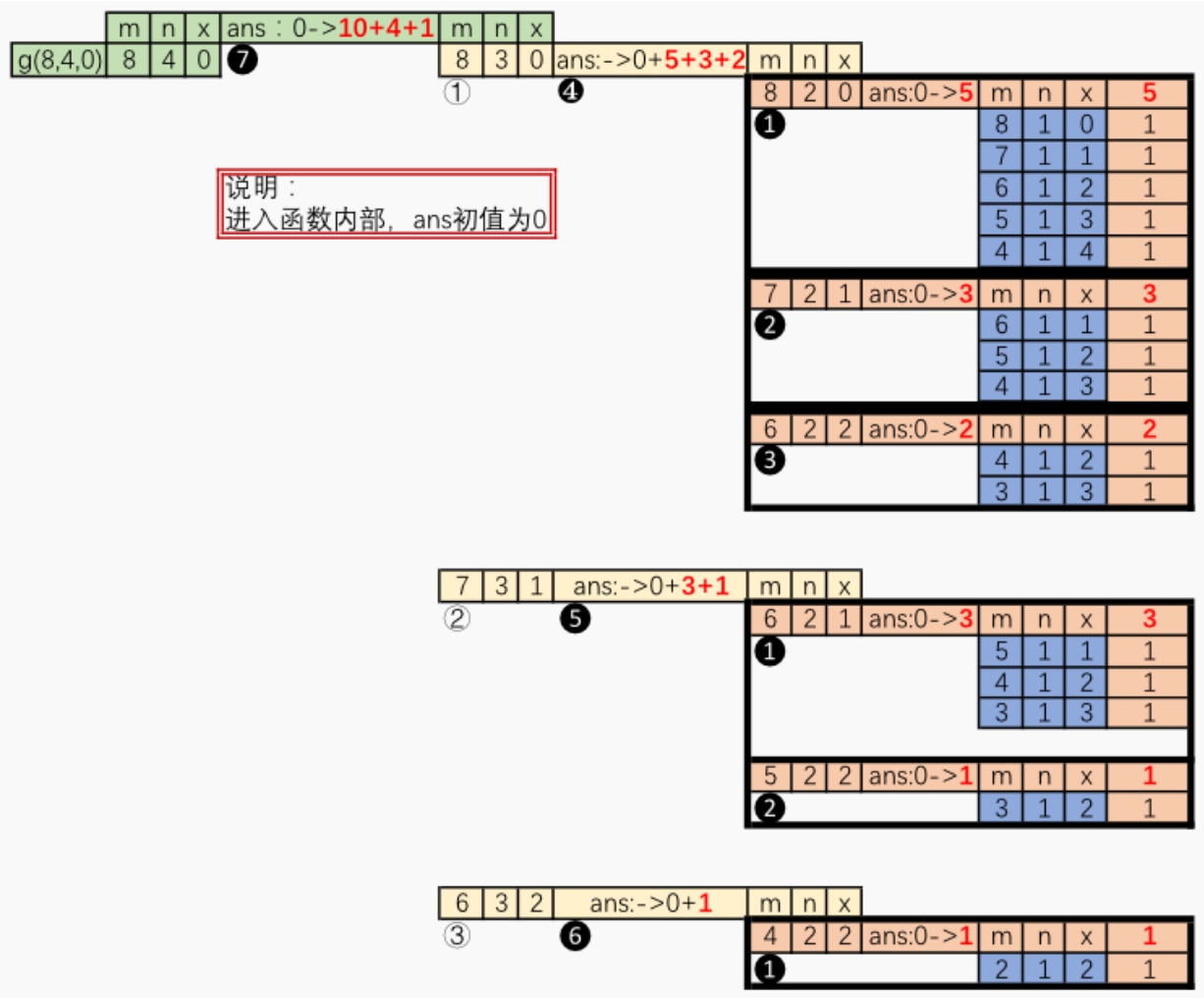
```
int g(int m, int n, int x) {
    int ans = 0;
    int i;
    if (n == 1)
        return 1;
    for (i = x; i <= m / n; i++)
        ans += g(m - i, n - 1, i);
return ans;
}

int main() {
    int t, m, n;
    cin >> m >> n;
    cout << g(m, n, 0) << endl;
    return 0;
}
```

输入：8 4 输出：_____

解析（简单题）：

递归程序运行及其结果，如下图所示：

答案是：15



2.
```
#include <iostream>

using namespace std;

int main() {
    int n, i, j, x, y, nx, ny;
    int a[40][40];
    for (i = 0; i < 40; i++)
        for (j = 0; j < 40; j++)
            a[i][j] = 0;
cin >> n;
y = 0; x = n - 1;
n = 2 * n - 1;
    for (i = 1; i <= n * n; i++){
        a[y][x] = i;
        ny = (y - 1 + n) % n;
        nx = (x + 1) % n;
        if ((y == 0 && x == n - 1) || a[ny][nx] != 0)
            y = y + 1;
       else { y = ny; x = nx; }
```

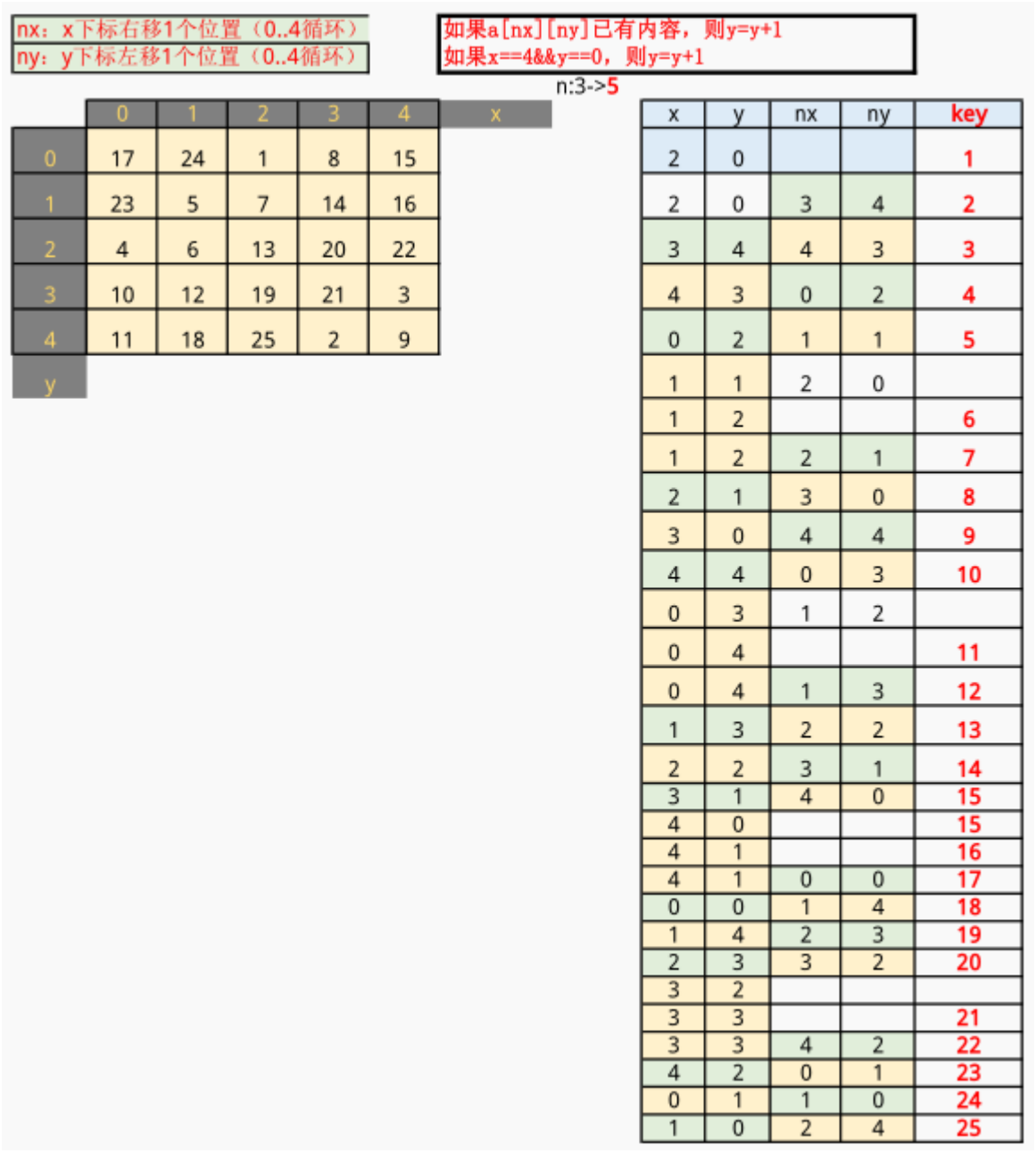
```
        }
        for (j = 0; j < n; j++)
           cout << a[0][j] << " ";
        cout << endl;
        return 0;
    }
```

输入：3

输出：_____

解析（简单题）：

找到幻方数据变化规律后，填写起来就舒服啦，没必要像我这样一一列举出来。



	0	1	2	3	4
0	17	24	1	8	15
1	23	5	7	14	16
2	4	6	13	20	22
3	10	12	19	21	3
4	11	18	25	2	9

x	y	nx	ny	key
2	0			1
2	0	3	4	2
3	4	4	3	3
4	3	0	2	4
0	2	1	1	5
1	1	2	0	
1	2			6
1	2	2	1	7
2	1	3	0	8
3	0	4	4	9
4	4	0	3	10
0	3	1	2	
0	4			11
0	4	1	3	12
1	3	2	2	13
2	2	3	1	14
3	1	4	0	15
4	0			15
4	1			16
4	1	0	0	17
0	0	1	4	18
1	4	2	3	19
2	3	3	2	20
3	2			
3	3			21
3	3	4	2	22
4	2	0	1	23
0	1	1	0	24
1	0	2	4	25

答案是：17　23　4　10　11

```
#include <iostream>
using namespace std;
int n, s, a[100005], t[100005], i;
void mergesort(int l, int r) {
   if (l == r)
      return;
int mid = (l + r) / 2;
int p = l;
int i = l;
int j = mid + 1;
mergesort(l, mid);
mergesort(mid + 1, r);
while (i <= mid && j <= r) {
      if (a[j] < a[i]) {
         s += mid - i + 1;
         t[p] = a[j];
         p++;
         j++;
      } else {
         t[p] = a[i];
         p++;
         i++;
       }
   }
   while (i <= mid) {
      t[p] = a[i];
      p++;
      i++;
}
while (j <= r) {
      t[p] = a[j];
      p++;
      j++;
}
for (i = l; i <= r; i++)
a[i] = t[i];
}

int main() {
      cin >> n;
      for (i = 1; i <= n; i++)
         cin >> a[i];
      mergesort(1, n);
      cout << s << endl;
      return 0;
   }
```

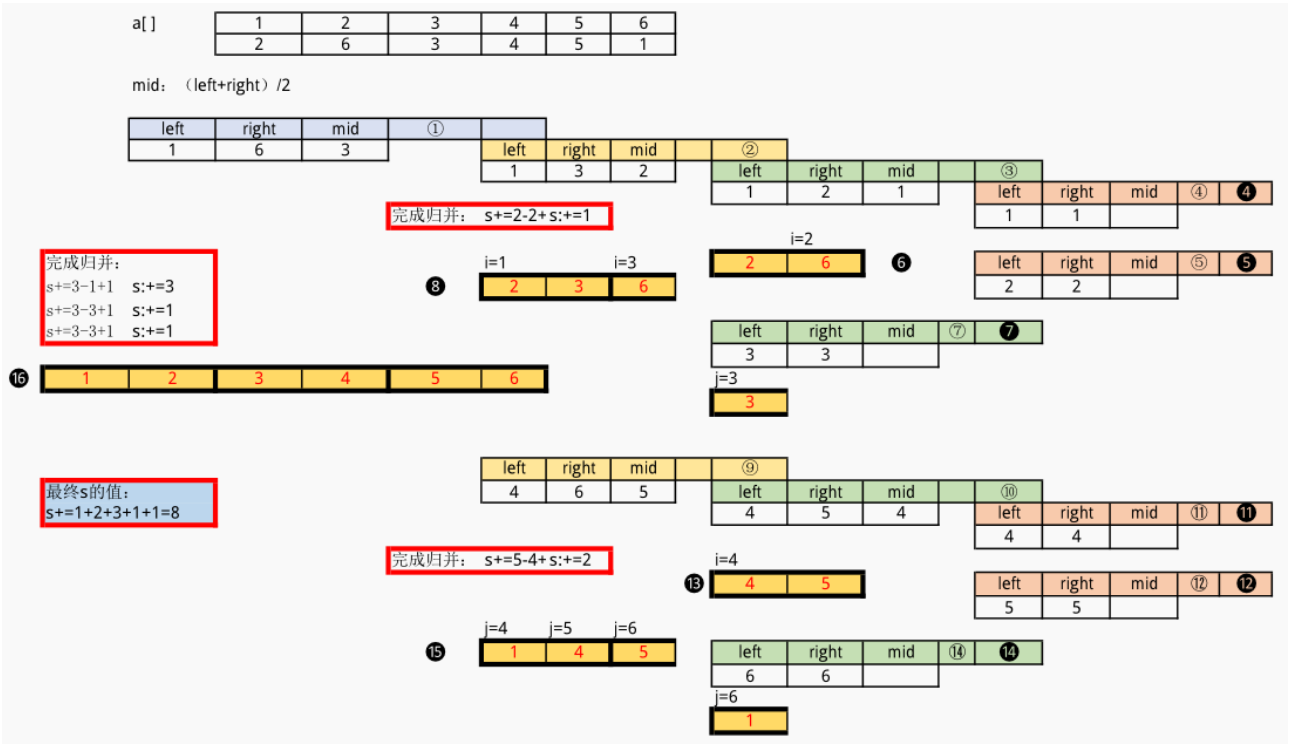
输入：6

2 6 3 4 5 1 输出：_____

解析（简单题）：

归并排序，数据量很小，直接模拟即可，重点观察变量 s 值的累加变化。

s 值的累加条件是：当比较时，后面区域元素较小时，则 s 累加。



答案是：8

4.
```
#include <iostream>

using namespace std;

int main() {
    int n, m;
    cin >> n >> m;
    int x = 1;
    int y = 1;
    int dx = 1;
    int dy = 1;
    int cnt = 0;
    while (cnt != 2) {
        cnt = 0;
        x = x + dx;
        y = y + dy;
        if (x == 1 || x == n) {
            ++cnt;
```

```
            dx = -dx;
        }
        if (y == 1 || y == m) {
            ++cnt;
            dy = -dy;
        }
    }
    cout << x << " " << y << endl;
    return 0;
}
```

输入 1：4 3
输出 1：_____（2 分）
输入 2：2017 1014
输出 2：_____（3 分）
输入 3：987 321
输出 3：_____（3 分）

解析（简单推理题）：

小数据模拟，观察找出数据变化规律。

1. 根据循环结束条件，变量 cnt 值必须等于 2 循环才结束，因此必须确保两条 if 语句条件均成立。也就是 x、y 须同时满足是 1……n、1……m 边界取值。

2. 找规律，先分析 x、y 的变化与 k 值的关系，列举出有效的 k 值。

左边 x 的变化，对应 k 值：4、7、10、13……，是一个等差数列：1+(n-1)*p(p>=1)

右边 y 的变化，对应 k 值：3、5、7、9……，是一个等差数列：1+(m-1)*q(q>=1)

示例分析：

输入 1：4　3

当 n=4 、m=3 时，k1=1+3*p、k2=1+2*q(p、q>=1)

要使 cnt 值为 2，必须同时符合边界值，因此有：

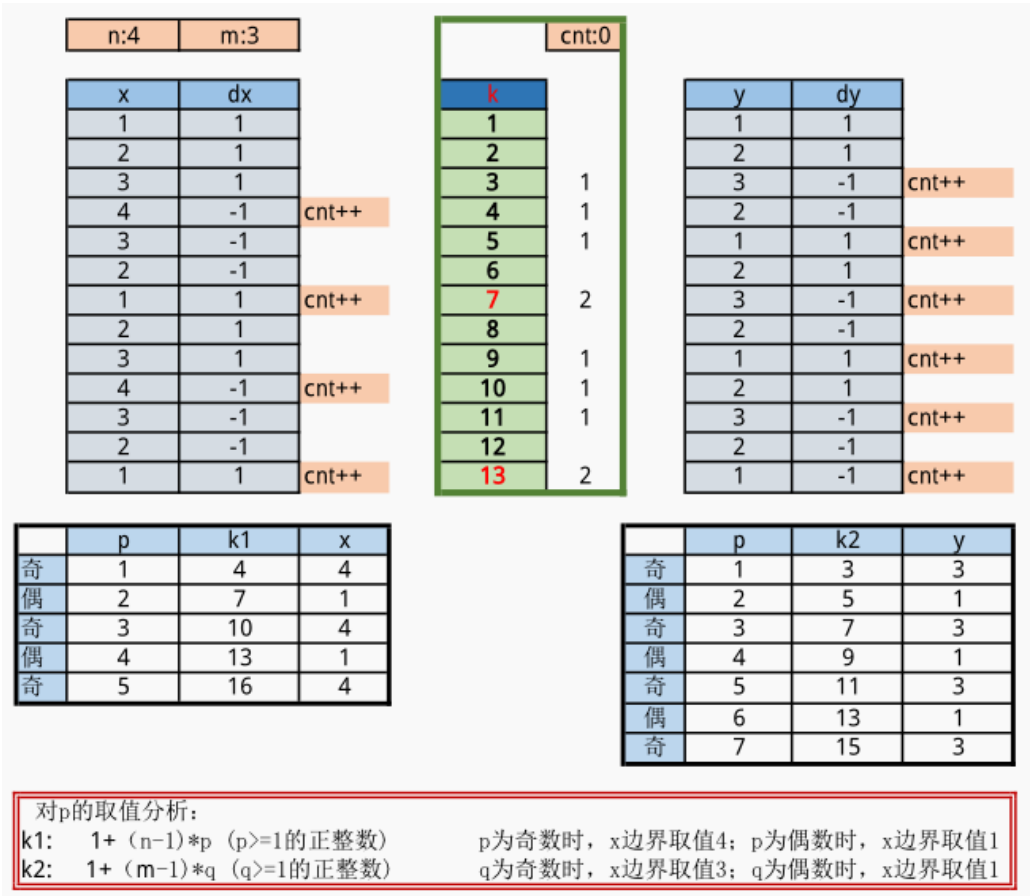
1+3p=1+2q(p、q>=1)，即：3p=2q(p、q>=1)

根据等式，求出同时整除 3、2 的第 1 个整数，即最小公倍数，LCM[3,2]= 6

找到第 1 个符合条件的 k：6+1=7，同时找出与之对应的 p 值(1+3p=7（p>=1）)，然后根据 p 的奇偶来确定边界值是开头、还是结束的数。

从图中，可知第 1 次出现 cnt==2 的情况是：x=1，y=3，

因此，输出 1：答案是 1　3



输入 2：2017 1014	输入 3：987 321
1+(2017-1)*p=1+(1013-1)*q（p、q>=1）	1+(986-1)*p=1+(320-1)*q（p、q>=1）
因此，LCM[2016, 1013]=2042208 2042208=2016*1013	因此，LCM[986, 320]=157760 157760=2*493*160
p 取值为 1013，p 为奇数，x 边界取 2017	p 取值为 160，p 为偶数，x 边界取 1
p 取值为 2016，p 为偶数，y 边界取 1	q 取值为 493，p 为奇数，y 边界取 321
因此，输出 2：答案是 2017 1	因此，输出 3：答案是 1 321

五、完善程序（共 **2** 题，每题 **14** 分，共计 **28** 分）

1.（大整数除法）给定两个正整数 p 和 q，其中 p 不超过 10^{100}，q 不超过 100000，求 p 除以 q 的商和余数。（第一空 2 分，其余 3 分）输入：第一行是 p 的位数 n，第二行是正整数 p，第三行是正整数 q。

输出：两行，分别是 p 除以 q 的商和余数。

```
#include <iostream>

using namespace std;

int p[100]; int n, i, q,
rest; char c;

int main() {
```

```
    cin >> n;
    for (i = 0; i < n; i++) {
       cin >> c;
       p[i] = c - '0';
    }
    //字符串“327659”除以837，将字符串转换成数值依次存储到整型数组p[ ]中
```

0	1	2	3	4	5	6	7	8
3	2	7	6	5	9			

```
    cin >> q;
    rest = ____(1)____;                    //p[0]
//在while循环体中第1次访问的是数组p[1]，因此p[0]用来当作rest的初值。
    i = 1;
    while (____(2)____ && i < n) {         //rest<q
        rest = rest * 10 + p[i];
        i++;
}
//在while循环体中最后访问的元素边界是：p[n-1]，rest在这里是进行依
//次访问数组p[ ]，并不断进行累乘，很明显是在试商，因此，此处填写rest<q。
if (rest < q)
       cout << 0 << endl;
else {
          cout << ____(3)____;
          //输出当前试商的第1位数字:          //rest/q
          //下面，逐位相乘，逐位试商，模拟除法过程
          while (i < n) {
             rest = ____(4)____;
             //下一位:                       //(rest % q)*p[i]
             i++;
             cout << rest / q;
          }
          cout << endl;
    }
    cout << ____(5)____ << endl;
    //最后输出高精度除以单精度数的余数:        //rest % q
        return 0;
}
```

解析（简单题，高精度简单模拟）：

高精度数据除以单精度数据，简单模拟即可以完成，想做错都难啊。

2. （最长路径）给定一个有向无环图，每条边长度为 1，求图中的最长路径长度。（第五空 2 分，其余 3 分）

输入：第一行是结点数 n（不超过 100）和边数 m，接下来 m 行，每行两个整数 a，b，表示从结点 a 到结点 b 有一条有向边。结点标号从 0 到(n-1)。

输出：最长路径长度。

提示：先进行拓扑排序，然后按照拓扑序计算最长路径。

```
#include <iostream>

using namespace std;

int n, m, i, j, a, b, head, tail, ans;

int graph[100][100];     // 用邻接矩阵存储图

int degree[100];         // 记录每个结点的入度

int len[100];            // 记录以各结点为终点的最长路径长度

int queue[100];          // 存放拓扑排序结果

int main() {
   cin >> n >> m;
   for (i = 0; i < n; i++)
      for (j = 0; j < n; j++)
         graph[i][j] = 0;
for (i = 0; i < n; i++)      // 存储图中每个结点的度数
      degree[i] = 0;
for (i = 0; i < m; i++) {
      cin >> a >> b;
      graph[a][b] = 1;         // 读入一条有向边 a->b
        ___(1)___;             // 结点 b 入度增 1: degree[b]++
   }
   // 拓扑排序，首先输出图中入度为 0 的结点，这里直接将入度为 0 结点入队列
   tail = 0;
   for (i = 0; i < n; i++)
      if (___(2)___) {      // 将入度为 0 的结点 i 入队 degree[i]==0
         queue[tail] = i;
         tail++;
      }
// 拓扑排序，从队列中逐个输出入度为 0 的结点，输出同时更新相关连结点度数
// head 为队列队首下标
head = 0;
```

```
while (tail < n - 1) {
        for (i = 0; i < n; i++)
        // 从队列中取队首结点 x，即结点编号：queue[head]
        // 若 graph[x][i]==1，说明结点 x 有边从 x 出发指向 i
        // 结点 x 将出队，扫描所有与 x 相关连结点 i，调整 x 指向结点 i 的入度
            if (graph[queue[head] ][i] == 1) {
                (3)      ;        // degree[i]--
                if (degree[i] == 0) {
        // 更新结点 i 的入度时发现结点 i 的入度为 0，将结点 i 进入队列
                    queue[tail] = i;
                    tail++;
                }
            }
            (4)      ;       // 队首元素出队 head++
    }
    ans = 0;
    for (i = 0; i < n; i++) {
        a = queue[i];
// 依照前面存储的拓扑顺对各个结点进行访问，同时记录到达当前结点 i 的最长路
径
        len[a] = 1;
        for (j = 0; j < n; j++)
            if (graph[j][a] == 1 && len[j] + 1 > len[a])
                len[a] = len[j] + 1;
        if (      (5)      ) // 结点 a 最长路径比 ans 值大，更新 ans ans<len[a]
            ans = len[a];
     }
    cout << ans << endl;
    return 0;
}
```

解析（简单题）：

依照题目，按照拓扑排序思路完成操作即可，不太容易丢分。