第 7 章 历年真题解析

7.1 计算机常识

NOIP2019

1. 中国的国家顶级域名是（ ）

A. .cn　　B. ,ch　　C. .chn　　D. .china

答案：A

解析：典型的国家顶级域名有.cn （中国）、.us （美国）、.uk（英国）、.jp（日本）、.sg （新加坡）等。

典型的通用顶级域名有.edu（教育机构）、.gov （政府部门）、.net（网络组织）、.com （商业组织）、.org（非营机构）、.mil （军事部门）等。

3. 一个 32 位整型变量占用（ ）个字节。

A. 32　　B. 128　　C. 4　　D. 8

答案：C

解析：8 位是 1 字节，因此 32 位是 4 字节。在 C++语言中，int 是最常用的带符号 32 位整型变量，可表示数值：$[-2^{31}, 2^{31}-1]$，unsigned int 是最常用的无符号 32 位整型变量，可表示数值$[0, 2^{32}-1]$。

15. 以下哪个奖项是计算机科学领域的最高奖？（ ）

A.图灵奖　B.鲁班奖　C.诺贝尔奖　D.普利策奖

答案：A

解析：图灵奖由美国计算机协会于 1966 年设立，其名称取自计算机科学之父图灵，专门奖励对计算机事业作出重要贡献的个人，被誉为“计算机界的诺贝尔奖” 。

NOIP2018

1. 以下哪一种设备属于输出设备：（ ）

A.扫描仪　B. 键盘　C. 鼠标　D. 打印机

答案：D

3. 1MB 等于（ ）。

A.1000 字节　　B. 1024 字节

C.1000 X 1000 字节　　D. 1024 X 1024 字节

答案：D

4. 广域网的英文缩写是（ ）。

A.LAN　B.WAN　C.MAN　D.LNA

答案：B

解析:WAN:Wide area network 广域网,LAN:local area network 局域网，MAN:Metropolitan Area Network 城域网。

5. 中国计算机学会于（ ）年创办全国青少年计算机程序设计竞赛。

A.1983 B.1984 C.1985 D.1986

答案：B

解析：考察比赛相关历史。1984 年邓小平指出:“计算机的普及要从娃娃做起。”教育部和中国科协委托中国计算机学会举办了全国青少年计算机程序设计竞赛（简称：NOI），1984 年参加竞赛的有 8000 多人。这一新的活动形式受到党和政府的关怀，得到社会各界的关注与支持。

NOIP2017

2. 计算机存储数据的基本单位是（ ）。

A. bit B. Byte C. GB D. KB

答案：B

3.下列协议中与电子邮件无关的是（ ）。

A. POP3 B. SMTP C. WTO D. IMAP

答案：C

解析：邮件相关的协议 SMTP（Simple Mail Transfer Protocol，简单邮件传输协议）、POP3（Post Office Protocol、邮局协议），IMAP（Internet Mail Access Protocol，Internet 邮件访问协议）

5. 计算机应用的最早领域是（ ）。

A. 数值计算 B. 人工智能 C. 机器人 D. 过程控制

答案：A

6.下列不属于面向对象程序设计语言的是（ ）。

A. C B. C++ C. Java D. C#

答案：A

7. NOI 的中文意思是（ ）。

A. 中国信息学联赛 B. 全国青少年信息学奥林匹克竞赛

C. 中国青少年信息学奥林匹克竞赛 D. 中国计算机协会

答案：B

18. 从（ ）年开始，NOIP 竞赛将不再支持 Pascal 语言。

A. 2020 B. 2021 C. 2022 D. 2023

答案：C

20. 以下和计算机领域密切相关的奖项是（ ）。

A. 奥斯卡奖 B. 图灵奖 C. 诺贝尔奖 D. 普利策奖

答案：B

NOIP2016

1.以下不是微软公司出品的软件是（ ）。

A. Powerpoint B. Word

C. Excel D. Acrobat Reader

答案：D

3. 以下不属于无线通信技术的是（ ）。

A. 蓝牙 B. WiFi C. GPRS D. 以太网

答案：D

4. 以下不是 CPU 生产厂商的是（ ）。

A. Intel B. AMD C. Microsoft D. IBM

答案：C

5. 以下不是存储设备的是（ ）。

A. 光盘 B. 磁盘 C. 固态硬盘 D. 鼠标

答案：D

9. 以下是 32 位机器和 64 位机器的区别的是（ ）。

A. 显示器不同 B. 硬盘大小不同

C. 寻址空间不同 D. 输入法不同

答案：C

20. 参加 NOI 比赛，以下不能带入考场的是（ ）。

A. 钢笔 B. 适量的衣服 C. U 盘 D. 铅笔

答案：C

NOIP2015

1. 1MB 等于（ ）。

A. 1000字节 B. 1024字节 C. 1000*1000 字节 D. 1024*1024 字节

答案：D

2.在 PC 机中，PENTIUM（奔腾）、酷睿、赛扬等是指（　　）。

A.生产厂家名称　B.硬盘的型号　C. CPU 的型号　D.显示器的型号

答案：C

3.操作系统的作用是（　　）

A.把源程序译成目标程序　　B.便于进行数据管理

C.控制和管理系统资源　　D.实现硬件之间的连接

答案：C

解析：操作系统是管理计算机硬件、软件资源，调度用户作业程序和处理各种中断，从而保证计算机各部分协调高效地工作的系统软件，它的功能是控制管理计算机系统资源和程序的执行。

4.在计算机内部用来传送、存贮、加工处理的数据或指令都是以（　）形式进行的。

A. 二进制码 B.八进制码　C.十进制码　D.智能拼音码

答案：A

5.下列说法正确的是（　　）。

A.　CPU的主要任务是执行数据运算和程序控制

B.　存储器具有记忆能力，其中信息任何时候都不会丢失

C.　两个显示器屏幕尺寸相同，则它们的分辨率必定相同

D.　个人用户只能使用Wifi的方式连接到Internet

答案：A

8.所谓的“中断”是指（　　）。

A.　操作系统随意停止一个程序的运行

B.　当出现需要时，CPU暂时停止当前程序的执行转而执行处理新情况的过程

C.　因停机而停止一个程序的运行

D.　电脑死机

答案：B

9.计算机病毒是（　　）。

A.　通过计算机传播的危害人体健康的一种病毒

B.　人为制造的能够侵入计算机系统并给计算机带来故障的程序或指令集合

C.　一种由于计算机元器件老化而产生的对生态环境有害的物质

D.　利用计算机的海量高速运算能力而研制出来的用于疾病预防的新型病毒

答案：B

10. FTP可以用于（　　）。

A.远程传输文件　B.发送电子邮件　C.浏览网页　D.网上聊天

答案：A

解析：FTP：File Transfer Protocol，文件传输协议。

11. 下面哪种软件不属于即时通信软件（　　）。

A. QQ　B. MSN　C.微信　D. P2P

答案：D

18. 下列选项中不属于视频文件格式的是（　　）。

A. TXT　B. AVI　C. MOV　D. RMVB

答案：A

20. 在 NOI 系列赛事中参赛选手必须使用由承办单位统一提供的设备。下列物品中不允许选手自带的是（　　）。

A.鼠标　B.笔　C.身份证　D.准考证

答案：A

7.2 基本运算

NOIP2019

2. 二进制数 11 1011 1001 0111 和 01 0110 1110 1011 进行逻辑与运算的结果是（　）。

A. 01 0010 1000 1011　B. 01 0010 1001 0011

C. 01 0010 1000 0001　D. 01 0010 1000 0011

答案：D

解析：逐位进行与运算。对于每一位，0 与 0 得 0，1 与 0 得 0，0 与 1 得 0，1 与 1 得 1。

NOIP2018

1. 下列四个不同进制的数中，与其它三项数值上不相等的是（　　）。

A. (269)16　B. (617)10　C. (1151)8　D.(1001101011)2

答案：D

NOIP2017

1. 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

答案：B

解析：反码：10101010，原码：11010101，对应十进制下的-85。

15. 十进制小数 13.375 对应的二进制数是（　　）。

A. 1101.011　　B. 1011.011　　C. 1101.101　　D. 1010.01

答案：A

解析：整数部分除 2 取余，小数部分乘 2 取整。

NOIP2016

7. 二进制数 00101100 和 00010101 的和是（　　）。

A. 00101000　　B. 01000001　　C. 01000100　　D. 00111000

答案：B

解析：参照十进制的加法运算，只是 2 进制加法要注意逢 2 进 1。

8. 与二进制小数 0.1 相等的八进制数是（　　）。

A. 0.8　　B.0.4　　C. 0.2　　D. 0.1

答案：B

解析：0.1 转 8 进制，需在小数点后添加 2 个 0，变为 0.100，再转 8 进制为：0.4。

NOIP2015

6. 二进制数 00100100 和 00010100 的和是（　　）。

A. 00101000　　B. 01000001　　C. 01000100　　D. 00111000

答案：D

解析：参照十进制的加法运算，只是 2 进制加法要注意逢 2 进 1。

7. 与二进制小数 0.1 相等的十六进制数是（　）

A. 0.8　　B. 0.4　　C. 0.2　　D. 0.1

答案：A

解析：2 进制转 16 进制，将每 4 位 2 进制转 1 位 16 进制，答案为$(0.1000)_2 = (0.8)_{16}$。

7.3 数据结构与算法

NOIP2019

5. 设有 100 个已排好序的数据元素，采用折半查找时，最大比较次数为（　）

A. 7　　B. 10　　C. 6　　D. 8

答案:A

解析：对 100 个有序元素进行折半查找，每次查找可将检索范围缩小一半。由 $2^6-1<100<=2^7-1$ 可知，最大比较次数为 7。对于 n 个数折半查找，最大比较次数是 $\log_2(n+1)$ 取上整。

6. 链表不具有的特点是（　）

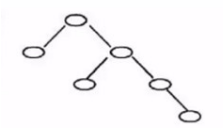
A. 插入删除不需要移动元素　　B. 不必事先估计存储空间

C. 所需空间与线性表长度成正比　　D. 可随机访问任一元素

答案：D

解析：链表是通过记录每个元素的后继位置来实现数据存储，所需空间与元素个数成正比，优点是不必事先估计存储空间、插入或删除指定位置元素的时间复杂 度为 O(1);但缺点是由于其元素的内存地址不连续，无法进行 O(1)的随机访问。

8. 一棵二叉树如右图所示，若采用顺序存储结构，即用一维数组元素存储该二叉树中的结点(根结点的下标为 1,若某结点的下标为 i，则其左孩子位于下标 2i 处、右孩子位于下标 2i + 1 处)，则该数组的最大下标至少为（ ）。



A. 6 B. 10 C. 15 D. 12

答案：C

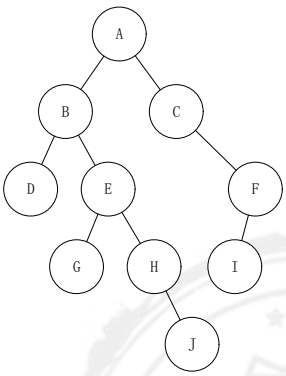
解析：根据题目给定的规则可知，下标最大的结点为树中深度最大且最靠右的结点,其下标为((1*2+1)*2+1)*2+1=15。

14. 假设一棵二叉树的后序遍历序列为 DGJHEBIFCA，中序遍历序列为 DBGEHJACIF， 则其前序遍历序列为（ ）。

A. ABCDEFGIIIJ B. ABDEGHJCFI C. ABDEGJHCFI D. ABDEGHJFIC

答案:B

解析：后序遍历的规则是“左右根”、中序遍历的规则是“左根右”，因此可知， A 是树根、DBGEHJ 是 A 左子树的中序遍历（对应后续遍历 DGJHEB）、CIF 是 A 右子树的中序遍历（对应后续遍历 IFC）,递归画出对应的二叉树，再根据前序遍历规则“根左右”即可求出答案。



NOIP2018

7.根节点深度为 0，一棵深度为 h 的满 k（k>1）叉树，即除最后一层无任何子节点外，每一层上的所有结点都有 k 个子结点的树，共有（ ）个结点。

A. $(k^{h+1} - 1) / (k - 1)$

B. k^{h-1}

C. k^{h}

D. $(k^{h-1}) / (k - 1)$

答案：A

解析：假设 h=2，k=2，画出完美二叉树，共 7 个节点，带入运算，得到选 A。

推导过程：

深度为 h 的树(根节点深度为 0),其总节点数为:

$s=k^0+k^1+k^2+\cdots+k^h$ （公式 1）

两边都乘以 k,得到:

$sk=k^1+k^2\cdots+k^h+k^{h+1}$ （公式 2）

公式 2 减公式 1,得到

$(k-1)s=k^{h+1}-k^0=k^{h+1}-1$

于是 $s=(k^{h+1}-1)/(k-1)$

8. 以下排序算法中，不需要进行关键字比较操作的算法是（　　）。

A. 基数排序　　B. 冒泡排序

C. 堆排序　　D. 直接插入排序

答案：A

解析：基数排序，根据键值的每位数字来分配桶；

冒泡排序、堆排序和插入排序都是基于两两元素关键字比较的排序。基数排序是直接将元素通过某些规则放到数组的相应位置，不是基于两两元素关键字比较的排序。

9. 给定一个含 N 个不相同数字的数组，在最坏情况下，找出其中最大或最小的数，至少需要 N - 1 次比较操作。则最坏情况下，在该数组中同时找最大与最小的数至少需要（　　）次比较操作。（⌈ ⌉表示向上取整，⌊ ⌋表示向下取整）

A. ⌈3N / 2⌉ - 2　　B. ⌊3N / 2⌋ - 2

C. 2N - 2　　D. 2N - 4

答案：A

解析：2 个数需要比较 1 次，可以筛选出答案在 AB 中，1 个数不需要比较，因此 BD 错误，答案选 A。

解析：

如果分别找最大值、最小值，则至少都需要 N-1 次操作。

同时找最大最小值，按照下面的优化算法：

N 为奇数时，比较次数为 3*(N-1)/2 =(3N+1)/2 - 2

N 为偶数时，比较次数为 1+3*(N-2)/2 = 3N/2 - 2

综合奇偶，显然答案为 A

找最大最小值的优化算法：

1、初始值：

2、N 为奇数，最大值、最小值的初始值都设为第一个元素。

3、N 为偶数，将前两个元素比较，最大值初始值为大的元素，最小值初始值为小的元素。

4、枚举，每次两个元素（循环步长为 2）

比较两个元素，分出大小。

大的元素与最大值比较，比最大值大则设为该元素。

小的元素与最小值比较，比最小值小则设为该元素。

5、循环结束，得到最大、最小值。

10. 下面的故事与（　　）算法有着异曲同工之妙。

从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事：“从前有座山，山里有座庙，庙里有个老和尚在给小和尚讲故事：‘从前有座山，山里有座庙，庙里有个老和尚给小和尚讲故事……’”

A. 枚举　B. 递归　　C. 贪心　　D. 分治

答案：B

领新教育青少年编程

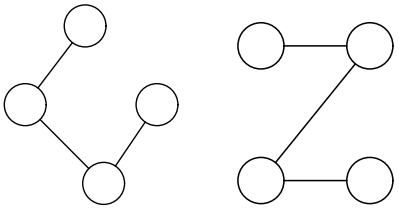
11. 由四个没有区别的点构成的简单无向连通图的个数是（　　）。

A. 6　　B. 7　　C. 8　　D. 9

答案：A

解析：简单无向图指的是，既不含平行边也不含环的图。

注意：下列两张图，其实是一种图。



因此，本题的答案有 6 种图，穷举的技巧是穷举边的数量，最少是 3 条边，最多是 6 条边（完全图）。

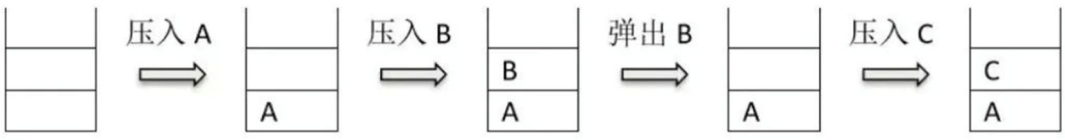
3 条边的情况：

4 条边的情况：

5 条边的情况：

6 条边的情况：

15. 下图中所使用的数据结构是（　　）。



A. 哈希表　　B. 栈　　C. 队列　　D. 二叉树

答案：B

NOIP2017

10. 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

答案：A

解析：m-(n-1)=m-n+1

13. 向一个栈顶指针为 hs 的链式栈中插入一个指针 s 指向的结点时，应执行（　　）。

A. hs->next = s;

B. s->next = hs; hs = s;

C. s->next = hs->next; hs->next = s;

D. s->next = hs; hs = hs->next;

答案：B

16. 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

答案：C

NOIP2016

10. 以下关于字符串的判定语句中正确的是（　　）。

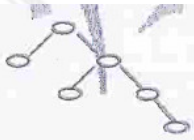
A. 字符串是一种特殊的线性表　　B. 串的长度必须大于零

C. 字符串不可以用数组来表示　　D. 空格字符组成的串就是空串

答案：A

解析：考察数据结构基础知识。空串的长度为零。字符串可以通过数组下标访问每一个元素。空格也是有效字符，由若干个空格组成的字符串也是有内容的。

11. 一棵二叉树如右图所示，若采用顺序存储结构，即用一 维数组元素存储该二叉树中的结点（根结点的下标为 1， 若某结点的下标为 i ，则其左孩子位于下标 2i 处、右孩 子位于下标(2i+1)处），则图中所有结点的最大下标为（　　）。



A. 6　　B. 10　　C. 12　　D. 15

答案：D

15. 设简单无向图 G 有 16 条边且每个顶点的度数都是 2，则图 G 有（　　）个顶点。

A. 10　B. 12　C. 8　D. 16

答案：D

解析：

抓住结点度数之和为边数的两倍来解题：

设顶点个数 x 个：

所以：2*x=16*2，x=16，所以答案选：D

NOIP2015

12. 6个顶点的连通图的最小生成树，其边数为（ ）

A. 6 B. 5 C. 7 D. 4

答案：B

解析：6 个顶点，边数=6-1=5。

13. 链表不具备的特点是（ ）

A. 可随机访问任何一个元素 B. 插入、删除操作不需要移动元素

C. 无需事先估计存储空间大小 D. 所需存储空间与存储元素个数成正比

答案：A

14. 线性表若采用链表存储结构，要求内存中可用存储单元地址（ ）

A.必须连续 B.部分地址必须连续 C. 一定不连续 D.连续不连续均可

答案：D

15. 今有一空栈 S，对下列待进栈的数据元素序列 a,b,c,d,e,f 依次进行进栈，进栈，出栈，进栈，进 栈，出栈的操作，则此操作完成后，栈 S 的栈顶元素为（ ）

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

答案：B

16. 前序遍历序列与中序遍历序列相同的二叉树为（ ）

A. 根结点无左子树的二叉树

B. 根结点无右子树的二叉树

C. 只有根结点的二叉树或非叶子结点只有左子树的二叉树

D. 只有根结点的二叉树或非叶子结点只有右子树的二叉树

答案：D

17. 如果根的高度为 1，具有 61 个结点的完全二叉树的高度为（ ）

A. 5 B. 6 C. 7 D. 8

答案：B

7.4 数学与逻辑

NOIP2019

7. 把 8 个同样的球放在 5 个同样的袋子里，允许有的袋子空着不放，问共有多少种不同的分法?（　　）提示:如果 8 个球都放在一个袋子里,无论是哪个袋子,都只算同一种分法。

A. 22　B. 24　C. 18　D. 20

答案：C

解析：数学计数问题，枚举法求解，8 个同样的球分 1 个袋子共 1 种方案，分 2 个袋子共 4 种方案，分 3 个袋子共 5 种方案，分 4 个袋子共 5 种方案，分 5 个袋子共 3 种方案，合计 18 种。

9. 100 以内最大的素数是（　　）。

A. 89　B. 97　C. 91　D. 93

答案：B

解析：98 - 100 均为合数，97 为素数。

10. 319 和 377 的最大公约数是（　　）。

A. 27　B. 33　C. 29　D. 31

答案：C

解析：使用辗转相除法可得 GCD（319, 377） =GCD（319, 58） =GCD（58, 29）=29。或将两数分解质因数后，提取公共部分亦可求解。

11. 新学期开学了，小胖想减肥，健身教练给小胖制定了两个训练方案。方案一： 每次连续跑 3 公里可以消耗 300 千卡（耗时半小时）：方案二：每次连续跑 5 公里可以消耗 600 千卡（耗时 1 小时）。小胖每周周一到周四能抽出半小时跑步，周五到周日能抽出一小时跑步。另外，教练建议小胖每周最多跑 21 公里，否则会损伤膝盖。请问如果小胖想严格执行教练的训练方案，并且不想损伤膝盖，每周最多通过跑步消耗多少千卡？（　　）

A. 3000 B. 2500 C. 2400 D. 2520

答案：C

解析：设方案 1 执行 x 天，方案 2 执行 y 天，则有：

3x+5y<=21，x+y<=7， x<=4，y<=3，要求 300x+600y 的最大值；

枚举可得最优方案为 x=2、y=3,此时 300x+600y 为 2400。或使用线性规划亦可求解。

x	y	300x+600y	3x+5y
1	1	900	8
1	2	1500	13
1	3	2100	18
2	1	1200	11
2	2	1800	16
2	**3**	**2400**	**21**
3	1	1500	14
3	2	2100	19
3	3	2700	24
4	1	1800	17
4	2	2400	22
4	3	3000	27

12. 一副纸牌除掉大小王有 52 张牌，四种花色，每种花色 13 张。假设从这 52 张 牌中随机抽取 13 张纸牌，则至少（ ）张牌的花色一致。

A. 4 B. 2 C. 3 D. 5

答案：A

解析：抽屉原理，最坏情况，13 张牌对应四种花色的牌数为 3、3、3、4。抽屉存储：黑桃黑桃黑桃，抽屉 2：红桃红桃红桃，抽屉 3：梅花梅花梅花，抽屉 4：方片方片方片 这时再抽一张，不管放在哪，都能满足要求。

13. 一些数字可以颠倒过来看，例如 0、1、8 颠倒过来还是本身，6 颠倒过来是 9，9 颠倒过来看还是 6，其他数字颠倒过来都不构成数字。类似的，一些多位数也可以颠倒过来看，比如 106 颠倒过来是 901。假设某个城市的车牌只由 5 位数字组成，每一位都可以取 0 到 9。请问这个城市最多有多少个车牌倒过来恰好还是原来的车牌？（ ）

A. 60 B. 125 C. 75 D. 100

答案：C

解析：排列组合题。枚举每位数字的可能性。颠倒后还得是个数字，因此前 2 位有 0,1,8,6,9，5 种选择，第 3 位只能放 0,1,8，后 2 位由前 2 位决定。总数=5*5*3*1*1=75。

NOIP2018

6. 如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 CapsLock、字母键 A、字母键 S、字母键 D、字母键 F 的顺序循环按键，即 CapsLock、A、 S、D、F、CapsLock、a、s、d、f、……，屏幕上输出的第 81 个字符是字母（ ）。

A. A B. S C. D D. a

答案：A

解析：考察字符串相关知识。按键顺序为 ASDFasdfASDFasdf...，周期为 8，所以 81 % 8 = 1，答案是 A。

12. 设含有 10 个元素的集合的全部子集数为 S，其中由 7 个元素组成的子集数为 T，则 T / S 的值为（ ）。

A. 5 / 32 B. 15 / 128 C. 1 / 8 D. 21 / 128

答案：B

解析：

解法一：

$T = C_{10}^{7} = C_{10}^{3} = 120$

$S = C_{10}^{0} + C_{10}^{1} + C_{10}^{2} + \cdots + C_{10}^{10} = 1024$

$T / S = 120 / 1024 = 15 / 128$

解法二：

考察排列与组合基础知识。10 个元素的集合的子集数有 2^10=1024 个，由 7 个元素组成

的子集数有 C(10，7)= C(10，3)= 120 种，故答案是 120 / 1024 = 15 / 128。

注意：

n 个元素的子集数：集合的子集可以含集合中的任意元素,甚至可以是空集,所以集合中的每个元素都可以有选或不选的可能.每个元素都有两个选择.含有 n 种元素的集合中,子集是 2*2*……*2 即 2 的 n 次方个。

13. 10000 以内，与 10000 互质的正整数有（ ）个。

A. 2000 B. 4000 C. 6000 D.8000

答案：B

解析：容斥原理，互质的意思，与 10000 没有公约数，即不能被 10000 的质因子整除。

10000 分解质因子：10000 =2*2*2*2*5*5*5*5

10000 以内被 2 整除的数有 5000 个

10000 以内被 5 整除的数有 2000 个

两者存在重复计算的数，即被 10 整除的数，有 1000 个。

被 2 或 5 整除的数有：5000 +2000 - 1000 =6000

互质的数有：10000 - 6000 = 4000 个

NOIP2017

4. 分辨率为 800x600、16 位色的位图，存储图像信息所需的空间为（ ）。

A. 937.5KB B. 4218.75KB C. 4320KB D. 2880KB

答案：A

解析：考察计算机科学基础知识。存储该图所需要的 bit 数为 16 * 800 * 600 = 7680000bit，1KB = 8 * 1024 = 8192bit，所需空间为 7680000 / 8192 = 937.57680000/8192=937.5 KB。

8. 2017 年 10 月 1 日是星期日，1999 年 10 月 1 日是（ ）。

A. 星期三 B. 星期日 C. 星期五 D. 星期二

答案：C

解析：考察基本的数学运算。对于非闰年来说，365 % 7 = 1，所以经过一年相当于在星期周期上往后走一天。对于闰年，365 % 7 = 2，相当于往后走两天。

闰年即能被 4 整除且不能被 100 整除的年份，或者能被 400 整除的年份。

1999 到 2017 有 18 年，5 个闰年（2000，2004，2008，2012，2016），13 个非闰年，相当于经过了（13 + 5 * 2） % 7 = 2 天，即从星期日倒推两天，是星期五。

9.甲、乙、丙三位同学选修课程，从 4 门课程中，甲选修 2 门，乙、丙各选修 3 门，则不同的选修方案共有（ ）种。

A. 36 B. 48 C. 96 D. 192

答案：C

解析：考察排列和组合基础知识。4 门课选修两门有 C(2/4)= 6 种，选修三门有 C(3/4) = 4 种，3 个同学的选课满足乘法原理，共有 6 * 4 * 4 = 96 种。

11. 对于给定的序列{ak}，我们把（i，j）称为逆序对当且仅当 i < j 且 ai > aj。那么序列 1，7，2，3，5，4 的逆序对数为（ ）个。

A. 4　　B. 5　　C. 6　　D. 7

答案：B

解析：概念理解题。根据概念，可以数出一共有（7，2），（7，3），（7，5），（7，4），（5，4）共 5 对逆序对，相当于求每个数的后面比自己小的数有多少个。

12. 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

答案：B

解析：考察数据结构基础知识。后缀表达式的计算规则是每次遇到操作数，将操作数加入栈里，每次遇到运算符 op，就将栈顶的前两个操作数弹出来进行 num1 op num2 的运算，可知选项 a b c + * d * 运算后符合条件。

14. 若串 S = “copyright”，其子串的个数是（ ）。

A. 72　　B. 45　　C. 46　　D. 36

答案：C

解析：考察字符串基础知识。

穷举法：长度为 9 有 1 种，长度为 8 有 2 种，长度为 7 有 3 种……长度为 1 有 9 种，这里的总数=1+2+3+4+5+6+7+8+9=45 种，不要忘记空子串，也就是长度为 0 的情况，因此结果是 46 种。

归纳：长度为 n 的字符串其子串个数 = n*(n+1)/2 + 1，注意空子串的情况。

插板取子串法：想象向 n 个字符中间插入两片木板，这两片木板之间的即为原串的一个子串。总共有 n+1 个空位可以插，第一个木板插入后，第二个还有 n 个空位。

所以共有 n(n+1) 种插法，又由于两片木板交换顺序后，子串还是同一个子串，所以子串数量应为 n(n+1)/2 。但最后，空串是任意字符串的子串，所以最后还要+1。

19. 一家四口人，至少两个人生日属于同一月份的概率是（　）（假定每个人生日属于每个月份的概率相同且不同人之间相互独立）。

A. 1/12　　B. 1/144　　C. 41/96　　D. 3/4

答案：C

解析：考察概率和排列组合知识。4 个人生日月份各不相同的概率是$\frac{A_{12}^4}{12^4}=\frac{55}{96}$，所以至少两个人生日月份相同的概率是 1 - 55/96 = 41/96。

NOIP2016

2. 如果 256 种颜色用二进制编码来表示，至少需要（　）位。

A. 6　B. 7　C. 8　D. 9

答案：C

6. 如果开始时计算机处于小写输入状态，现在有一只小老鼠反复按照 CapsLock、 字母键 A、字母键 S 和字母键 D 的顺序循环按键，即 CapsLock、A、S、D、CapsLock、a、s、d、……，屏幕上输出的第 81 个字符是字母（ ）。

A. A B. S C. D D. a

答案：C

解析：按照按键顺序，显示的内容为：ASDasdASDasd...，每 6 个字母一组，因此第 81 个字母之前，已经有 13 组共 78 个字母显示出来了，第 79 个字母为 A，第 80 个字母为 S，第 81 个字母为 D。

16. 有 7 个一模一样的苹果，放到 3 个一样的盘子中，一共有（ ）种放法。

A. 7 B. 8 C. 21 D. 37

答案：B

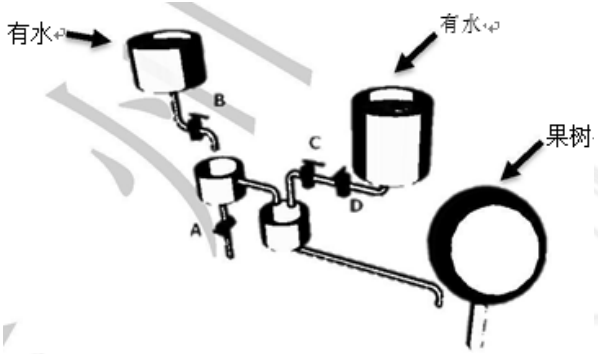
考察数学中的排列组合知识。可以枚举所有可能情况：

三个盘子有两个盘子没放苹果：所有苹果放在一个盘子里，1 种放法；

三个盘子中有一个没放苹果：由于苹果和盘子都一模一样，因此有 3 种放法（1，6）、（2、5）、（3、4）；

三个盘子都放了苹果：有 4 种放法（1，1，5）、（1，2，4）、（1，3，3）、（2，2，3）一共 8 种情况。

17. 下图表示一个果园灌溉系统，有 A、B、C、D 四个阀门，每个阀门可以打开 或关上，所有管道粗细相同，以下设置阀门的方法中，可以让果树浇上水的是（ ）。



A. B 打开，其他都关上 B. AB 都打开，CD 都关上

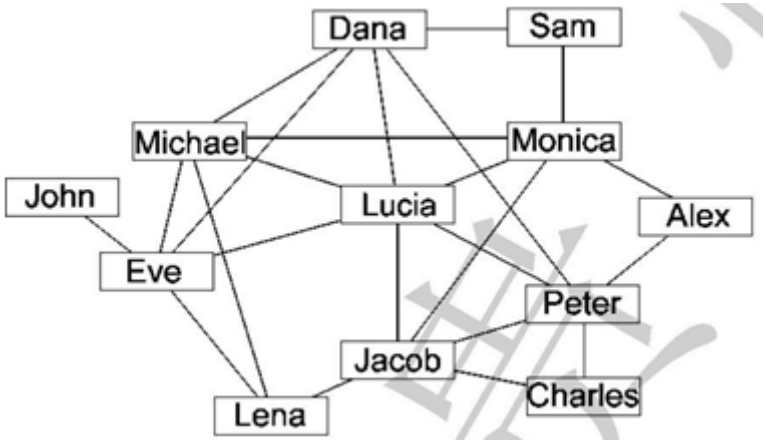
C. A 打开，其他都关上 D. D 打开，其他都关上

答案：A

解析：常识题。只有 C、D 同时打开，或 B 打开的同时 A 关闭，才有可能浇水。

18. Lucia 和她的朋友以及朋友的朋友都在某社交网站上注册了账号。下图是他们之间的关系图，两个人之间有边相连代表这两个人是朋友，没有边相连代表不是朋友。这个社交网站的规则是：如果某人 A 向他（她）的朋友 B 分 享了某张照片，那么 B 就可以对该照片进行评论；如果 B 评论了该照片，那 么他（她）的所有朋友都可以看见这个评论以及被评

论的照片，但是不能对该照片进行评论（除非 A 也向他（她）分享了该照片）。现在 Lucia 已经上传了一张照片，但是她不想让 Jacob 看见这张照片，那么她可以向以下朋友（　）分享该照片。



A. Dana, Michael, Eve　　B. Dana, Eve, Monica

C. Michael, Eve, Jacob　　D. Micheal, Peter, Monica

答案：A

解析：考察图论相关知识，也是一道较容易理解的常识题。由于 Lucia 不想让 Jacob 看到照片，她分享的这些朋友需要满足一个条件，就是他们不能有 Jacob 这个人作为朋友，反映到图上就是：与 Lucia 直接相连但不与 Jacob 直接相连的结点，是满足要求的。

19. 周末小明和爸爸妈妈三个人一起想动手做三道菜。小明负责洗菜、爸爸负责切菜、妈妈负责炒菜。假设做每道菜的顺序都是：先洗菜 10 分钟，然后切 菜 10 分钟，最后炒菜 10 分钟。那么做一道菜需要 30 分钟。注意：两道不 同的菜的相同步骤不可以同时进行。例如第一道菜和第二道的菜不能同时洗，也不能同时切。那么做完三道菜的最短时间需要(　) 分钟。

A. 90　　B. 60　　C. 50　　D. 40

答案：C

解析：

时间	10	20	30	40	50	60	70	80	90
洗	1	2	3						
切		1	2	3					
炒			1	2	3				

7.5 程序设计

NOIP2019

4. 若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值（c 大于 0）

```
s = a;
for （b = 1; b <= c; b++） s = s - 1;
```

则与上述程序段功能等价的赋值语句是（ ）

A. s = a - c;　　B. s = a - b;　　C. s = s - c;　　D. s = b - c;

答案：A

解析：s 初始化为 a，紧接着 for 循环 c 次，每次 s 减 1，因此该程序段相当于 s=a-c。

NOIP2018

14. 为了统计一个非负整数的二进制形式中 1 的个数，代码如下：

```
int CountBit(int x)
{
    int ret = 0;
    while (x)
    {
        ret++;
        __________;
    }
    return ret;
}
```

则空格内要填入的语句是（ ）。

A. x >>= 1

B. x &= x - 1

C. x |= x >> 1

D. x <<= 1

答案：B

解析：>>：右移运算符，5>>1，将 5 换算为二进制 101，右移 1 得到 10，转为 10 进制是 2，因此右移相当于除 2，<<相当于乘 2。

a&b：将 a 和 b 分别转换为二进制进行&运算，比如：5&4 = (101) & (100) = (100) = 4。

a|b，同理，转为二进制用|运算。

NOIP2017

17. 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，任何以元素比较作为基本运算的归并算法在最坏情况下至少要做（ ）次比较。

A. n^2　　B. n log n　　C. 2n　　D. 2n - 1

答案：D

解析：考察算法基础知识。归并排序当有一方为空的时候，就可以不用再进行比较，直接将另一方剩余元素接在数组后面即可。否则每进行一次比较，均有一个元素可以接到数组后面，故总比较次数为：总元素个数减去一方为空时另一方的剩余元素个数。

所以最少的剩余元素个数是 1，答案为比较次数为 2n-1 次。

NOIP2016

12. 若有如下程序段，其中 s、a、b、c 均已定义为整型变量，且 a、c 均已赋值 （c 大于 0）。

```
s = a;
for (b = 1; b <= c; b++) s = s + 1;
```

则与上述程序段修改 s 值的功能等价的赋值语句是（ ）。

A. s = a + b;　B. s = a + c;　C. s = s + c;　D. s = b + c;

答案：B

解析：考察程序设计基础知识。先将 s 赋值为 a；再循环 c 次，每次给 s 累加 1，相当于给 s 一共累加了 c。总体看来，是将 s 赋值为 a + c。

13. 有以下程序：

```
#include <iostream>
using namespace std;
int main() {
    int k = 4, n = 0;
    while (n < k) {
        n++;
        if (n % 3 != 0)
            continue;
            k--;
    }
    cout << k << "," << n << endl;
    return 0;
}
```

程序运行后的输出结果是（　　）。

A. 2,2　　B. 2,3　　C. 3,2　　D. 3,3

答案：D

解析：考察程序设计基础知识。当 n 不是 3 的倍数时，n++，并且 k--，当 n = 3，k = 3 时，会跳出循环。类似题目，可以手动模拟一遍程序运行过程，记录中间过程变量的值。

14. 给定含有 n 个不同的数的数组 L=<x_1, x_2, ..., x_n>。如果 L 中存在 x_i（1 < i < n） 使得 $x_1 < x_2 < ... < x_{i-1} < x_i > x_{i+1} > ... > x_n$， 则称 L 是单峰的，并称 x_i 是 L 的“峰顶”。现在已知 L 是单峰的，请把 a-c 三行代码补全到算法中使得算法 正确找到 L 的峰顶。

```
a.  Search(k+1, n)
b.  Search(1, k-1)
c.  return L[k]
Search(1, n)
1. k<-[n/2]
2. if L[k] > L[k-1] and L[k] > L[k+1]
3. then __________
4. else if L[k] > L[k-1] and L[k] < L[k+1]
5. then __________
6. else __________
```

正确的填空顺序是（ ）。

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

答案：A

解析：考察算法基础知识。利用二分法寻找 “峰顶” 。

如果第 k 个数大于第 k - 1 个数和第 k + 1 个数，那么 k 就是峰顶；

如果第 k 个数大于第 k - 1 个数，但小于第 k + 1 个数，那么峰顶在 k 的右侧，需要在 k + 1 到 n 的范围内去寻找峰顶；否则，应该去 1 到 k + 1 的范围内去寻找峰顶。

NOIP2015

19. 设某算法的计算时间表示为递推关系式 T(n)=T(n-1)+n(n 为正整数）及 T(0) = 1，则 该算法的时间复杂度为（ ）。

A. O(logn) B. O(n log n) C. O(n) D. $O(n^2)$

答案：D

解析：考察算法分析基础知识。

T(n)=T(n-1)+n

=T(n-2)+(n-1)+n

=1+2+3+...+n=n*(n+1)/2

故答案是 $O(n^2)$。

7.6 问题求解

NOIP2018

1、甲乙丙丁四人在考虑周末要不要外出郊游。

已知①如果周末下雨，并且乙不去，则甲一定不去；②如果乙去，则丁一定去；③如果丙去，则丁一定不去；④如果丁不去，而且甲不去，则丙一定不去。

如果周末丙去了，则甲_____（去了/没去）(1 分)，乙_____（去了/没去）(1 分)，丁_____（去了/没去）(1 分)，周末_____（下雨/ 没下雨）(2 分)。

解析：

考察逻辑推理基础知识。首先丙去了，根据 ④ 得知甲去了或者丁去了，又根据 ③ 得知丁不去，所以甲一定去了。

由于丁不去，根据 ② 得知乙不去。

由于甲去了，并且乙不去，根据 ① 得周末没下雨。

2. 从 1 到 2018 这 2018 个数中，共有_____个包含数字 8 的数。包含数字 8 的数是指有某一位是“8”的数， 例如“2018”与“188”。

答案：544

穷举，考察数学知识。

解法一：计算不含 8 的数字个数，用 2018-这个值得到结果。

1 位数不含 8 = 8 个

两位数不含 8 = 8*9 个（第 1 位有 8 种可能，第 2 位有 9 种可能）

三位数不含 8 = 8*9*9 个

[1000~2000)不含 8：9*9*9 个

>=2000 不含 8：19-2=17 个

总计 = 1474 个

解法二：

个位数是 8 的个数是 2018/10=201 加上 2018 这个数就是 202 个。

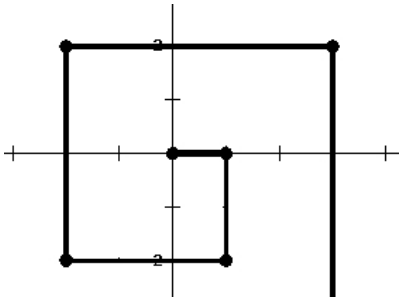
十位数是 8 个位数不是 8 的个数是 2018/100*9=180 个。

百位数是 8，十位个位都不是 8 的是 2018/1000*81=162 个。

所以总数有 202+180+162=544 个。

NOIP2017

1. 一个人站在坐标（0, 0）处，面朝 x 轴正方向。第一轮，他向前走 1 单位距离，然后右转；第二轮，他向前走 2 单位距离，然后右转；第三轮，他向前走 3 单位距离，然后右转……他一直这么走下去。请问第 2017 轮后，他的坐标是：（_____，_____）。（请在答题纸上用逗号隔开两空答案）



答：1009, 1008

解析：找规律，每一个循环有 4 个动作，上、下、左、右！

初始值为 0, 0，不看坐标的值，看坐标的变化；

第 1 循环

	x	y
	+1	0
	0	-2
	-3	0
	0	+4
变化	-2	+2

第 2 循环

	x	Y
	+5	0
	0	-6
	-7	0
	0	+8
变化	-2	+2

可以看出，每一循环的 x 和 y 的变化幅度一定是相同的，由此可以直接计算出

第 1 循环结束，坐标为-2, 2

第 2 循环结束，坐标为-4, 4

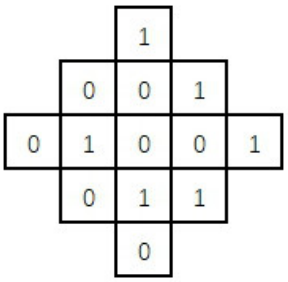
2017 次走完时，经过了 2017/4=504 次循环，余数为 1，说明是 504 次循环+1 次。

504 次循环结束，坐标应当为：504*-2，504*2，也就是-1008, 1008

第 504 次循环结束值到了 504*4=2016，下一个数是 2017，也就是下一步为+2017, 0

坐标变化为 1009, 1008。

2. 如下图所示，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。



答案：3

考察枚举的能力。显然一次不可以，接着通过枚举发现两次也不可以，三次可以。对第一行的格子、第三行第 3，4 个格子操作即可。

NOIP2016

1. 从一个 4×4 的棋盘（不可旋转）中选取不在同一行也不在同一列上的两个方格，共有_____种方法。

答案：72

解析：考察排列组合知识。选第一个点有 16 种可能，选第二个点时要与第一个点不同行不同列，共有 9 种可能。

由于第一次选方格 A 第二次选方格 B，第一次选方格 B 第二次选方格 A，这两种方案是一样的，最终共有 16 * 9 / 2 = 72 种方法。

2. 约定二叉树的根节点高度为 1。一棵结点数为 2016 的二叉树最少有_____个叶子结点；一棵结点数为 2016 的二叉树最小的高度值是_____。

答案：填空位置 ①：1，填空位置 ②：11

考察二叉树相关知识。一棵二叉树最少有 1 个叶子节点。

要使二叉树的高度尽可能小，需要保证所有非叶子节点尽可能都有左右孩子。一个高度值为 10 的完全二叉树最多包含 1023 个结点，一个高度值为 11 的完全二叉树最多包含 2047 个结点。因此结点数位 2016 的二叉树最小高度值为 11。

NOIP2015

1.重新排列1234使得每一个数字都不在原来的位置上，一共有_____种排法。

答案：9

解析：考察排列与组合基础知识。枚举全排列可知一共有：2143，2341，2413，3142，3412，3421，4123，4312，4321 这 9 种排法。

2.一棵结点数为2015的二叉树最多有_____个叶子结点。

答案：1008

解析：考察二叉树相关知识。二叉树的叶子节点个数 = 度为 2 的非叶子结点个数 +1，且可以使所有非叶子结点度都为 2,如完全二叉树。故叶子结点最多的情况为 n+(n-1)=2015，n=1008。

7.7 阅读程序

解题步骤讲解：

1、读懂程序

整体来说，能读懂程序的“目的”是最容易解题的；

2、通读程序

A、一般来说，面对有难度的程序，未必能完全理解程序的作用，那么可以先通读程序，了解程序大概做了哪些事情；

B、通读时面对一些不太理解的步骤，不要太担心，第一遍只是大概了解，心态在考试中也很重要；

C、阅读题干，了解题目问了哪些问题，带着问题再次阅读程序；

3、两种情况：

A、能彻底读懂程序的目的，那么解题会很方便；

B、**不能彻底读懂程序的目的，可以举例带入模拟**，通过不同的数据将程序模拟几遍，看看分别得到什么结果，但模拟直接要大概知道程序有哪些步骤，每个步骤再干嘛；

4、解题注意

A、模拟是很重要的解题方法；

B、学会将数据代入程序（例证），将问题代入程序；

C、模拟后尝试寻找、总结规律；

D、**拿分最关键，不理解算法的目的也能拿分**；

E、注意变量名、函数名，对题目也能起到“解释”作用；

F、递归题模拟前，先猜大意；

NOIP2019

二、阅读程序（程序输入不超过数组或字符串定义的范围；判断题正确填 √，错误填×；除特殊说明外，判断题 1.5 分，选择题 4 分，共计 40 分）

程序一

```
#include <cstdio>
#include <cstring>
using namespace std;
char st[100];
int main() {
  scanf("%s", st);
  int n = strlen(st);
  for (int i = 1; i <= n; ++i) {
     if (n % i == 0) {
       char c = st[i - 1];
       if (c >= 'a')
         st[i - 1] = c - 'a' + 'A';
     }
  }
  printf(st);
  return 0;
}
```

解析：程序用于将字符串中，如果第 i 个字符是 n 约数，且是 ascii 码>=’a’的字母转换为大写。

•判断题

1） 输入的字符串只能由小写字母或大写字母组成。（ ）

答案：错

解析：输入的字符串也可以包含数字等其他字符。

2） 若将第 8 行的“i = 1”改为“i = 0”，程序运行时会发生错误。（ ）

答案：对

解析：若 i 可以为 0,则第 9 行的 if 语句条件"n%i==0”将发生运行时错误 RE。

3） 若将第 8 行的“i <= n”改为“i * i <= n”，程序运行结果不会改变。（ ）

答案：错

解析：当第 8 行的循环条件为“i<=n”时，字符串的末尾字符会被程序加工，但若改为“i*i<=n”，字符串的末尾字符将不会被程序加工（除非字符串长度为 1）。

4） 若输入的字符串全部由大写字母组成，那么输出的字符串就跟输入的字符串一样。（ ）

答案：对

解析：大写字母的 ASCII 编码值小于小写字母的。若输入的字符串全部由大写字母组成，则程序不会对其进行加工。

•选择题

5） 若输入的字符串长度为 18，那么输入的字符串跟输出的字符串相比，至多有（ ）个字符不同。

A. 18　　B. 6　　C. 10　　D. 1

答案：B

解析：18 的正约数共有 6 个，因此程序至多修改输入字符串中的 6 个字符，即输出字符串与输入字符串至多有 6 个字符不同。

6） 若输入的字符串长度为（ ），那么输入的字符串跟输出的字符串相比，至多有 36 个字符不同。

A. 36　　B. 100000　　C. 1　　D. 128

答案:B

解析：根据程序的作用可知，要使输出字符串和输入字符串之间至多有 36 个字符不同，36 应当是字符串长度 n 的约数个数。本题选项中，仅有 100000 满足要求，将其分解质因数得 $100000=2^5*5^5$ 得其的正约数共有（5+1）*（5+1）=36 个。

程序二

```
#include <cstdio>
using namespace std;
int n, m;
int a[100],b[100];

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

假设输入的 n 和 m 都是正整数，x 和 y 都是在[1, n]的范围内的整数，完成下面的判断题和单选题：

解析：程序可以视作通过模拟算法选出一系列互不冲突的数对——若数对(x_1, y_1)和(x_2, y_2)之间有 $x_1==x_2$ 或 $y_1==y_2$ 则认为这两个数对存在冲突，现按顺序考虑每一个数对(x_i, y_i)（要求满足前提：在已经选用的数对中，与左值 x_i 匹配的右值小于 y_i，、与右值 y_i 匹配的左值小于 x_i），若该数对与此前已经选用的数对冲突， 则用当前数对替换所冲

突的原数对；若无冲突，则直接选用当前数对。最后的 ans 用于统计剩余多少值为 0，相当于统计没有相应数对被我们选中的值。

- **判断题**

1）当 m>0 时，输出的值一定小于 2n。（　　）

答案：对

解析：由限定条件 0<x,y<=n 可知，当 m>0 时，一定存在某个数对被我们选中,此时 ans<2n。

2）执行完第 27 行的“++ans”时，ans 一定是偶数。（　　）

答案：错

解析： ans 最终一定是偶数。但第 27 行的“++ans”在第 23 行的 for 循环的内部，其中间结果可能为奇数。

验证：假设 m=1,输入：1 2，在 i=1 时，ans 就是奇数。

3） a[i]和 b[i]不可能同时大于 0。（　　）

答案：错

解析：当存在数对(i,y)和(x, i)都被我们选中时，a[i]和 b[i]就会同时大于 0。比如：假设 m=1,输入：2 2，那么是可能同时为 0 的，再比如：m=2，输入两对数：1 3,3 1。

4） 若程序执行到第 13 行时，x 总是小于 y,那么第 15 行不会被执行。

答案：错

解析：存在反例——依次考虑数对(1,2) (1,3)时，第 15 行程序会被执行。

•选择题

5） 若 m 个 x 两两不同，且 m 个 y 两两不同，则输出的值为（　　）

A. 2n-2m　　B. 2n+2　　C. 2n-2　　D. 2n

答案：A

解析：此时，输入的数对两两互不冲突，因此程序会将它们全部选中，根据上述 ans 的意义可知，其结果为 2n-2m。

6） 若 m 个 x 两两不同，且 m 个 y 都相等，则输出的值为（　　）

A. 2n-2　　B. 2n　　C. 2m　　D, 2n-2m

答案:A

解析：此时，输入的数对两两存在冲突，因此程序最终只会选用一个数对，根据上述 ans 的意义可知，其结果为 2n-2。

程序三

```
1  #include <iostream>
2  using namespace std;
3  const int maxn = 10000;
```

```
int n;
int a[maxn];
int b[maxn];
int f(int l, int r,int depth){
  if (l > r)
    return 0;
  int min = maxn,mink;
  for (int i = l; i <= r; ++i) {
    if (min > a[i]) {
      min = a[i];
      mink = i;
    }
  }
  int lres = f(l,mink - 1, depth + 1);
  int rres = f(mink + 1,r, depth + 1);
  return lres + rres + depth * b[mink];
}
int main() {
  cin >> n;
  for (int i = 0; i < n; ++i)
    cin >> a[i];
  for (int i = 0; i < n; ++i)
    cin >> b[i];
  cout << f(0, n - 1, 1) << endl;
  return 6;
}
```

解析：程序可以视作根据二叉树的中序遍历，构造一棵满足要求的树，并输出各结点深度与 b 值的加权和——要求二叉树的根结点最小，并递归要求左右子树的根结点最小(除非相应子树为空)。(**注：不能完全看出程序的“目的”并不影响拿分!!!**)

•判断题

1）如果 a 数组有重复的数字，则程序运行时会发生错误。(　　)

答案：错

解析:若 a 数组有重复数字，特别是有多个最小值，默认取最左侧的最小，模拟可以发现可以有重复。

2）如果 b 数组全为 0,则输出为 0。(　　)

答案：对

解析：根据程序可以发现，计算结果由 depth * b[mink]决定，当 b 数组为 0，结果为 0。

•选择题

3） 当 n=100 时，最坏情况下，与第 12 行的比较运算执行的次数最接近的是:（ ）。

A. 5000 B. 600 C. 6 D. 100

答案:A

解析：最坏情况下，二分失效，比如 a 数组元素递增，第 1 次循环 100 个数，第 2 次循环 99 个，……最后一次循环 1 个数，因此执行次数 = 100 + 99 + … + 1，最接近的值是 A。

4） 当 n=100 时，最好情况下，与第 12 行的比较运算执行的次数最接近的是:（ ）。

A. 100 B. 6 C. 5000 D. 600

答案：D

解析：最佳情况下，恰好每次都能完全二分，那么执行次数为 nlogn≈600。

5） 当 n=10 时，若 b 数组满足，对任意 0 ≤ i < n。都有 b[i] = i + 1，那么输出最大为（ ）。

A. 386 B. 383 C. 384 D. 385

答案：D

解析：要使输出的 ans 值尽可能大，深度就要尽可能的深，也就是“最坏的完全不能二分的情况”，比如 a 数组的值恰好递增（递减也可以，但递减不能使得 ans 最大），此时 ans 的最大值为 1*1+2*2+3*3+…+10*10=385。

6） （4 分）当 n=100 时，若 b 数组满足，对任意 0 ≤ i < n，都有 b[i]=1，那么输出最小为（ ）。

A. 582 B. 580 C. 579 D. 581

答案：B

解析：此时，要使输出的 ans 值尽可能小，那么深度要尽可能的小，也就是完全能二分的情况，那么类似一个“完全二叉树”，这个树有 100 个节点，就有 7 层，前 6 层共有 63 个数，第 7 层有 37 个数，此时 ans 的值 = （1*1+2*2+3*4+…+6*32）+7*37=580。

每一个深度元素的个数：1, 2, 4, 8, 16, 32, 37。

NOIP2018

1.
```
#include <cstdio>
char st[100];
int main() {
  scanf("%s", st);
  for (int i = 0; st[i]; ++i) {
    if ('A' <= st[i] && st[i]<= 'Z')
       st[i] += 1;
 }
 printf("%s\n",st);
  return 0;
```

```
}
```

输入：QuanGuoLianSai

输出：_____

答案：RuanHuoMianTai

解析：字符+1。

2.
```
#include <cstdio>
int main() {
   int x;
   scanf("%d", &x);
   int res = 0;
   for (int i = 0; i < x; ++i) {
     if (i * i % x == 1) {
      ++res;
     }
   }
   printf("%d", res);
   return 0;
}
```

输入：15

输出：_____

答案：4

解析：要求 0~14 中，有几个数的平方 mod 15==1

3.
```
#include <iostream>
using namespace std;
int n, m;
int findans(int n, int m) {
    if (n == 0) return m;
    if (m == 0) return n % 3;
    return findans(n-1, m) - findans(n, m-1) + findans(n-1, m-1);
}
int main(){
    cin >> n >> m;
    cout << findans(n, m) << endl;
   return 0;
}
```

输入：5 6　输出：_____

答案：8

f(n,m)=f(n-1,m)-f(n,m-1)+f(n-1,m-1)

n=0 return m

m=0 return n%3

本题可以通过列表法，列一个二维表格，以 n 从小到大，m 从小到大的顺序依次算出每个格子的值。

格子（n，m）的值可以通过格子（n-1，m），（n，m-1）和（n-1，m-1）的值得到，从而得到（5，6）的值。

n/m	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	0	3	2	5	4	7
2	2	-1	4	1	6	3	8
3	0	1	2	3	4	5	6
4	1	0	3	2	5	4	7
5	2	-1	4	1	6	3	8

4.
```
#include <cstdio>
int n, d[100];
bool v[100];
int main() {
   scanf("%d", &n);
   for (int i = 0; i < n; ++i) {
          scanf("%d", d + i);
          v[i] = false;
    }
   int cnt = 0;
    for (int i = 0; i < n; ++i) {
       if (!v[i]) {
            for (int j = i; !v[j]; j = d[j])
            {
                 v[j] = true;
            }
            ++cnt;
```

```
        }
      }
      printf("%d\n", cnt);   return 0;
    }
```

输入：10 7 1 4 3 2 5 9 8 0 6

输出：_____

答案：6

解析：

d 数组：

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

v 数组：

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

发现输入先读入点的个数，再读入 d 数组， d 数组是个 0 到 9 这 10 个点的排列。d 数组的第 i 个值就是 i 出边的另一端点，程序求的是这个图中环的个数，共 6 个。

环分别是（0，7，8），（1），（2，4），（5），（6，9）。

NOIP2017

1.
```
#include <iostream>
using namespace std;
int main() {
   int t[256];
   string s;
   int i;
   cin >> s;
   for (i = 0; i < 256; i++)
        t[i] = 0;
   for (i = 0; i < s.length(); i++)
        t[s[i]]++;
   for (i = 0; i < s.length(); i++)
       if (t[s[i]] == 1) {
          cout << s[i] << endl;
          return 0;
       }
   cout << "no" << endl;
   return 0;
}
```

输入：xyzxyw

输出：_____

答案：z

解析：程序先读入字符串"xyzxyw"，并统计每个字母出现的次数，然后枚举字符串中的每个字母，如果该字母在字符串中只出现过一次，就输出该字母，并 return 0 结束程序。"xyzxyw" 中 z 和 w 都只出现过一次，但是由于 z 在前面，所以输出 z，程序结束。

2.
```
#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：7 3

输出：_____

答案：8

解析：可以直接模拟递归程序的计算过程：

g（7，3，0）= g（7，2，0）+ g（6，2，1）+ g（5，2，2）。

分别展开每一项：

g（7，2，0）= g（7，1，0）+ g（6，1，1）+ g（5，1，2）+ g（4，1，3）= 1 + 1 + 1 + 1 = 4。

g（6，2，1）= g（5，1，1）+ g（4，1，2）+ g（3，1，3）= 1 + 1 + 1 = 3。

g（5，2，2）= g（3，1，2） = 1。

所以答案是 4 + 3 + 1 = 8。

3.
```
#include <iostream>
using namespace std;
int main() {
   string ch; int a[200]; int b[200];
   int n, i, t, res;
   cin >> ch;
   n = ch.length();
```

```
    for (i = 0; i < 200; i++)
        b[i] = 0;
    for (i = 1; i <= n; i++) {
       a[i] = ch[i - 1] - '0';
       b[i] = b[i - 1] + a[i];
    }
    res = b[n];
    t = 0;
    for (i = n; i > 0; i--) {
      if (a[i] == 0)
          t++;
      if (b[i - 1] + t < res)
          res = b[i - 1] + t;
    }
    cout << res << endl;
    return 0;
}
```

输入：10011010110011011010111110001

输出：_____

答案：11

解析：这题需要细心的一步一步模拟程序的步骤得到结果。其中，a 是一个 0 和 1 的数组，b 是 a 的前缀和，a 数组的下标 1 对应字符串的 0 位置。

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：_____（3 分）

输入 2：2017 1014

输出 2：_____（5 分）

答案：

输出 1：1 3

输出 2：2017 1

解析：程序中：

x 按照 1,2,3,4,...,n,n-1,n-2,...,3,2,1,2,3...变化；

y 按照 1,2,3,4,...,m,m-1,m-2,...,3,2,1,2,3...变化；

每当 x 为 1 或者 n，y 为 1 或者 m 的时候会使 cnt 增加 1，并且每次移动前都会将 cnt 重新置为 0，所以答案一定是 x 或者 y 第一次同时触碰边界的时候。

同时触碰边界只有 4 种情况（左，左），（左，右），（右，右）和（右，左），可以列方程求解四种情况第一次出现的步数，最早出现的就是答案。

第一组例子是 x = 1，y = 3 的时候，为 6 步。

第二组例子是 x = 2017, y = 1 的时候，2042208 步。

NOIP2016

1.
```
#include <iostream>
using namespace std;
int main() {
  int max, min, sum, count = 0;
  int tmp;
  cin >> tmp;
  if (tmp == 0)
     return 0;
  max = min = sum = tmp;
  count++;
  while (tmp != 0) {
    cin >> tmp;
    if (tmp != 0) {
      sum += tmp;
      count++;
      if (tmp > max) max = tmp;
      if (tmp < min) min = tmp;
    }
```

```
    }
    cout << max << "," << min << "," << sum / count << endl;
    return 0;
}
```

输入：1 2 3 4 5 6 0 7

输出：_____

题目答案：6,1,3

题目解析：

如果读入的第一个数是 0，会直接结束程序；

如果后续读入的数字是 0，就会结束 while 循环；

在 while 循环内部，每次循环将读入的数累加到 sum 上，count 增加 1 记录读入的数字个数，并且更新最大值 max 和最小值 min。

因此对于本题的输入，读到 0 时就会输出内容并结束，此时 max 的值为 6，min 为 1，sum 为 21，count 为 3。需要注意 sum 和 count 均为整数，因此 sum / count 为整除，结果是 3。

输出结果为：6,1,3。

2.
```
#include <iostream>
using namespace std;
int main() {
    int i = 100, x = 0, y = 0;
    while (i > 0) {
        i--;
        x = i % 8;
        if (x == 1) y++;
    }
    cout << y << endl;
    return 0;

}
```

输出：_____

答案：13

题目解析：i 的初始值是 100，进入 while 循环之后，首先 i--，第一个被处理的数值是 99，同理最后一个被处理的数值是 0。

同时当 i 对 8 取模的结果是 1 时，y 增加 1。

因此本题是统计 99 到 0 之间有多少除以 8 余数为 1 的数字。数目不多，甚至可以枚举出来：97，89，81，73，65，57，49，41，33，25，17，9，1，一共 13 个。

3.
```
#include <iostream>
    using namespace std;
    int main() {
```

```
    int a[6] = {1, 2, 3, 4, 5, 6};
    int pi = 0;
    int pj = 5;
    int t , i;
    while (pi < pj) {
        t = a[pi];
        a[pi] = a[pj];
        a[pj] = t;
        pi++;
        pj--;
    }
    for (i = 0; i < 6; i++)
      cout << a[i] << ",";
    cout << endl;
    return 0;
}
```

输出：_____

答案：6, 5, 4, 3, 2, 1,

解析：本题可以通过列表法，在每一次循环中，更新每个变量中的值。不难发现，本题代码是将一个长度为 6 的数组前后颠倒。

4.

```
#include <iostream>
using namespace std;
int main() {
    int i, length1, length2;
    string s1, s2;
    s1 = "I have a dream." s2 = "I Have A Dream."; length1 =
    s1.size();
    length2 = s2.size();
    for (i = 0; i < length1; i++)
        if (s1[i] >= 'a' && s1[i] <= 'z')
            s1[i] -= 'a' - 'A';
    for (i = 0; i < length2; i++)
        if (s2[i] >= 'a' && s2[i] <= 'z')
            s2[i] -= 'a' - 'A';
    if (s1 == s2)
        cout << "=" << endl;
    else if (s1 > s2)
        cout << ">" << endl;
    else
        cout << "<" << endl;
    return 0;
}
```

输出：_____

答案：=

解析：题目中的两个 for 循环，会将 s1 和 s2 中所有小写字母都转换成大写字母，因此转换结束后，两个字符串是相等的。

NOIP2015

1.
```
#include <iostream>
using namespace std;
int main() {
    int a,b,c;
    a=1; b=2; c=3;
    if (a > c){
    if(a>c)
    cout << a <<" ";
    else cout << b <<" ";
    }
    cout << c << endl;
    return 0;
}
```

输出：_____

答案：3

解析：当执行到第一个 if 语句时，由于 a>b 不成立，所以整个大括号中的语句都不执行，直接执行 cout << c << endl; 输出变量 c 的值，即输出 3。

2.
```
#include <iostream>
using namespace std;
struct point {
    int x;
    int y;
};

int main() {
    struct EX {
        int a; int b;
        point c;
    } e;

    e.a=1; e.b=2;
    e.c.x = e.a + e.b;
    e.c.y = e.a * e.b;
    cout << e.c.x << ","<< e.c.y <<endl;
    return 0;
}
```

输出：_____

答案：3,2

解析

e.c.x = e.a + e.b; 即 e.c.x = 1 + 2 = 3;

e.c.y = e.a * e.b; 即 e.c.y = 1 * 2 = 3;

所以输出 3,2。

3.
```
#include <iostream>
#include <string>
using namespace std;
int main() {
  string str;
  int i;
  int count;
  count = 0;
  getline(cin,str);
  for (i = 0; i < str.length(); i++) {
      if(str[i] >= 'a' && str[i] <= 'z')
          count++;
  }
  cout << "It has " << count << " lowercases" << endl;
  return 0;
}
```

输入：NOI2016 will be held in Mian Yang.

输出：_____

答案：It has 18 lowercases

解析：

阅读程序可以发现，程序先把一整行字符串（包括空格）读进来，然后统计字符串中小写字母的总个数，输出语句也提示了程序的功能.

数一下输入的句子，发现一共有 1818 个小写字母，即输出语句中 count 为 1818。

4.
```
#include <iostream>
 using namespace std;
 void fun(char *a, char *b) {
     a = b;
     (*a)++;
 }
 int main() {
     char c1, c2, *p1, *p2;
```

```
    c1 = 'A';
    c2 = 'a';
    p1 = &c1;
    p2 = &c2;
    fun(p1, p2);
    cout << c1 << c2 << endl;
    return 0;
}
```

输出：_____

答案：Ab

解析：

p1 和 p2 都是字符型指针，一开始 p1 指向 c1，p2 指向 c2，调用 fun 函数之后将 p1 和 p2 两个指针的值（c1，c2 的地址传了进去），fun 函数里对 a 和 b 的修改不改变主函数 p1，p2 的值。

fun 函数中，b 指针的值先赋值给了 a，此时 a 指向了 c2 的地址，接着对 a 指针所指地址的值进行 +1 操作，故最后 c1 的值不变，c2 的值被加了 1，输出 Ab。

7.8 完善程序

NOIP2019

三、完善程序（单选题，每题 3 分，共计 30 分）

程序一

1.(矩阵变幻)有一个奇幻的矩阵，在不停的变幻，其变幻方式为：数字 0 变成矩阵$\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$，数字 1 变成矩阵$\begin{bmatrix}1 & 1\\1 & 0\end{bmatrix}$，最初该知阵只有一个元素 0，变幻 n 次后，矩阵会变成什么样？

例如，矩阵最初为：[0]；矩阵变幻 1 次后：$\begin{bmatrix}0 & 0\\0 & 1\end{bmatrix}$；矩阵变幻 2 次后：

$$\begin{bmatrix}0 & 0 & 0 & 0\\0 & 1 & 0 & 1\\0 & 0 & 1 & 1\\0 & 1 & 1 & 0\end{bmatrix}$$

输入一行一个不超过 10 的正整数 n。输出变幻 n 次后的矩阵。

试补全程序。

提示：“<<”表示二进制左移运算符，例如$(11)_2<<2 = (1100)_2$；

而“^”表示二进制异或运算符，它将两个参与运算的数中的每个对应的二进制位——进行比较，若两个二进制位相同，则运算结果的对应二进制位为 0，反之为 1。

```
1    #include <cstdio>
2    using namespace std;
```

```
3     int n;
4     const int max_size = 1 << 10;
5     int res[max_size][max_size];
6     void recursive(int x,int y,int n,int t) {
7       if (n == 0) {
8         res[x][y]= ① ;
9         return;
10      }
11      int step = 1 << (n - 1);
12      recursive(②,n - 1, t);
13      recursive(x, y + step, n - 1, t);
14      recursive(x + step, y, n - 1, t);
15      recursive(③, n - 1, !t);
16    }
17
18    int main() {
19      scanf("%d", &n);
20      recursive(0,0,④);
21      int size = ⑤ ;
22      for (int i = 0; i < size; ++i) {
23        for (int j = 0; j < size; ++j)
24          printf("%d", res[i][j]);
25        puts("");
26      }
27      return 0;
28    }
```

解析：程序采用分治算法，递归模拟矩阵的变换过程。递归函数 recursive(x, y, n, t)表示计算左上角(x, y),大小 2^n*2^n 由单个数字 t 变幻而来的矩阵。

1）①处应填（　　）

A. n % 2　　B. 0　　C. t　　D. 1

答案:C

解析：此处为递归边界，当需要计算的是单位矩阵时，相应元素应赋值为 t,即无需再经任何变换。

2）②处应填（　　）

A. x - step , y - step　　B. x, y - step

C. x - step , y　　D. x, y

答案：D

解析：左上角（x,y）,且大小 2^n*2^n 的矩阵，可以分成 4 个 $2^{n-1}*2^{n-1}$ 的矩阵分别计算。此处需要计算的是 4 个矩阵中位于左上方的矩阵，该矩阵的左上角坐标为(x, y)。

3）③处应填（　　）

A. x - step, y - step　　B. x + step, y + step

C. x - step, y　　D. x, y - step

答案：B

解析：左上角（x,y）,且大小 2^n*2^n 的矩阵，可以分成 4 个 $2^{n-1}*2^{n-1}$ 的矩阵分别计算。此处需要计算的是 4 个矩阵中位于右下方的矩阵，该矩阵的左上角坐标为 （$x+2^{n-1}$, $y+2^{n-1}$）。

4） ④处应填（）

A. n - 1, n % 2　　B. n, 0

C. n, n % 2　　D. n - 1, 0

答案：B

解析：此处是递归计算的入口，即题目最终所求的是大小 2^n*2^n,由单个数字 0 变幻而来的矩阵，因此递归函数的后两个参数应设为 n 和 0。

5）⑤处应填（）

A. 1 << (n + 1)　　B. 1 << n

C. n + 1　　D. 1 << （n - 1）

答案：B

解析：此处是计算最终所求的矩阵大小，即边长 size 为 2^n,位运算写做“1<<n”

程序二

2.(计数排序)计数排序是一个广泛使用的排序方法。下面的程序使用双关键字计数排序，将 n 对 10000 以内的整数，从小到大排序。

例如有三对整数(3,4)、(2,4)、(3,3),那么排序之后应该是 (2,4)、(3,3)、(3,4)。

输入第一行为 n,接下来 n 行，第 i 行有两个数 a[i]和 b[i],分别表示第 i 对整数的第一关键字和第二关键字。

从小到大排序后输出。

数据范围 $1\leqslant n\leqslant 10^7$，$1\leqslant a[i],b[i]\leqslant 10^4$。

提示：应先对第二关键字排序，再对第一关键字排序。数组 ord[]存储第二关键字排序

的结果，数组 res[]存储双关键字排序的结果。

试补全程序。

```
#include <cstdio>
#include <cstring>
using namespace std;
const int maxn = 10000000;
const int maxs = 10000;
int n;
unsigned a[maxn], b[maxn], res[maxn], ord[maxn];
unsigned cnt[maxs + 1];

int main() {
    scanf("%d",&n);
    for (int i - 0; i < n; ++i)
        scanf("%d%d", &a[i], &b[i]);
    memset(cnt,0,sizeof(cnt));
    for (int i = 0; i < n; ++i)
        ① ; //利用 cnt 数组统计数量
    for (int i = 0; i < maxs; ++i)
        cnt[i + 1] += cnt[i];
    for (int i = 0; i < n; ++i)
        ② ; //记录初步排序结果
    memset(cnt, 0 , sizeof(cnt));
    for (int i = 0; i < n; ++i)
        ③ ; //利用 cnt 数组统计数量
    for (int i = 0; i < maxs; ++i)
        cnt[i + 1] += cnt[i];
    for (int i = n - 1; i >= 0; --i)
        ④ ; //记录最终排序结果
    for (int i = 0; i < n; ++i)
        printf("%d %d\n", ⑤);
    return 0;
}
```

解析：基于计数排序，程序实际实现了近似于基数排序的算法——依次以低位到高位的每一位数为关键词，进行计数排序，前一次的排序结果是下一次的初始序列—

—本题对应着先根据第二关键词 b 进行计数排序，再根据第一关键词 a 进行计数排序。

1) ①处应填(　　)

A. ++cnt[i]

B. ++cnt[b[i]]

C. ++cnt[a[i] * maxs + b[i]]

D. ++cnt[a[i]]

答案：B

解析：此处是根据第二关键词 b 进行计数排序，并做各关键词的数量统计工作，因此将 b[i]对应的元素数量自增 1。

2) ②处应填（　　）

A. ord[--cnt[a[i]]] = i

B. ord[--cnt[b[i]]] = a[i]

C. ord[--cnt[a[i]]] = b[i]

D. ord[--cnt[b[i]]] = i

答案：D

解析:此处是根据第二关键词 b 进行计数排序，并记录排序结果。此时的 ent [key]用于表示关键词为 key 的元素在结果数组中的位置，因此这里的程序应将关键词为 b[i]的元素 i 放在 ord 数组里。

3) ③处应填（）

A. ++cnt[b[i]]

B. ++cnt[a[i] * maxs + b[i]]

C. ++cnt[a[i]]

D. ++cnt [i]

答案:C

解析：此处是根据第一关键词 a 进行计数排序，并做各关键词的数量统计工作，因此将 a[i]对应的元素数量自增 1。

4) ④处应填（）

A. res[--cnt[a[ord[i]]]] = ord[i]

B. res[--cnt[b[ord[i]]]] = ord[i]

C. res[--cnt[b[i]]] = ord[i]

D. res[--cnt[a[i]]] = ord[i]

答案：A

解析：此处是根据第一关键词 a 进行计数排序，并记录排序结果。由于此前已经根据第二关键词 b 进行计数排序，此时第 i 个元素的原始下标实际为 ord[i]，因此这里的程序应将关键词为 a[ord[i]]的元素 ord[i]放在 res 数组里。

5） ⑤处应填（）

A. a[i], b[i]

B. a[res[i]], b[res[i]]

C. a[ord[res[i]]] , b[ord[res[i]]]

D. a[res[ord[i]]] , b[res[ord[i]]]

答案：B

解析：此处是按顺序输出排序结果。由于此前已经按照第二关键词、第一关键词完成了计数排序，此时第 i 个元素的原始下标实际为 res[i]。

NOIP2018

1. （最大公约数之和）下列程序想要求解整数n的所有约数两两之间最大公约数的和对 10007 求余后的值，试补全程序。（第一空 2 分，其余 3 分）

举例来说，4 的所有约数是 1,2,4。1 和 2 的最大公约数为 1；2 和 4 的最大公约数为 2；1 和 4 的最大公约数为 1。于是答案为 $1+2+1=4$。

要求 getDivisor 函数的复杂度为O函数 $(\sqrt{n})$, gcd 的复杂度为$O(\log \max(a,b))$。

```
#include <iostream>
using namespace std;
const int N = 110000, P = 10007;
 int n;  int a[N], len;
 int ans;
 void getDivisor() {
    len = 0;
    for (int i = 1; ____(1)____ <= n; ++i)
     if (n % i == 0) {
      a[++len] = i;
      if (____(2)____ != i) a[++len] = n / i;
     }
   }
   int gcd(int a, int b) {
        if (b == 0) {  ____(3)____;
    }
    return gcd(b, ____(4)____);
   }

   int main() {
     cin >> n;   getDivisor();  ans = 0;
    for (int i = 1; i <= len; ++i) {
       for (int j = i + 1; j <= len; ++j) {
```

```
            ans = (     (5)     ) % P;
          }
        }
        cout << ans << endl; return 0;
      }
```

答案：

（1）i*i　（2）n / i　（3）return a　（4）a % b　（5）ans + gcd(a[i],a[j])

枚举约数的时候，需要枚举到 n，所以第①空需要填 i * i，i * i <= n 就是枚举到 sqrt(n)。

如果 n % i == 0，i 和 n / i 都是 n 的约数，第 ② 空填 n / i 用于防止得到重复的约数。

第 ③④ 空位于辗转相除法求最大公约数函数里，分别填入 a 和 a % b。

第 ⑤ 空位于枚举约数，求两个约数的最大公约数并更新答案的地方，所以需要填上 ans + gcd(a[i], a[j])。

2. 对于一个 1 到n的排列P（即 1 到n中每一个数在P中出现了恰好一次），令q_i为第i个位置之后第一个比P_i值更大的位置，如果不存在这样的位置，则$q_i = n + 1$。 举例来说，如果$n = 5$且P为 1 5 4 2 3，则q为 2 6 6 5 6。

下列程序读入了排列P，使用双向链表求解了答案。试补全程序。（第二空 2 分，其余 3 分）

数据范围 $1 \leqslant n \leqslant 10^5$。

```
#include <iostream>
using namespace std;
const int N = 100010;
 int n;  int L[N], R[N],
a[N];
 int main() {
        cin >> n;
        for (int i = 1; i <= n; ++i) {
           int x;
            cin >> x;
            (1) ;
       }
        for (int i = 1; i <= n; ++i) {
            R[i] = (2) ;
            L[i] = i - 1;
        }
        for (int i = 1; i <= n; ++i) {
            L[ (3) ] = L[a[i]];
            R[L[a[i]]] = R[ (4) ];
```

```
        }
        for (int i = 1; i <= n; ++i) {
          cout << ____(5)____ << " ";
        }
        cout <<
        endl;
        return 0;
}
```

答案：

(1) a[x] = i　(2) i+1　(3) R[a[i]]　(4)a[i]　(5)R[i]

考察双向链表的基本操作。程序按照 x 的值从 1 到 n 枚举每个值，依次在双向链表中删除该点，删除的时候就可以知道该值左边最近比它大元素的位置和右边最近比它大元素的位置。

第 ① 空是 a 数组的初始化操作，标记每个值的位置，所以需要填 a[x] = i。

第 ② 空是双向链表指针的初始化操作，右指针指向右边相邻的元素，所以需要填 i + 1。

第 ③④ 空是双向链表的删除操作，当前元素右边元素的左指针需要指向左边元素，左边元素的右指针需要指向右边元素，所以分别填 R[a[i]] 和 a[i]。

最后需要输出答案，即 R[i] 的值，所以第 ⑤ 空需要填 R[i]。

解析 2：

1.a[i]表示大小为 i 的元素所在位置。

2.对称地写下一行的东西。链表的下一个元素。

3.对称地写下一行的东西。删除链表中本元素，把本元素右边元素的左指针跨过本元素指向左侧元素。

4.还是对称。把左边元素右指针指向右侧元素。

5.输出第 i 个数的后继元素即第一个比他大的元素，即 R[i]。

输入：1 5 4 2 3

a 数组：

	1	4	5	3	2	
0	1	2	3	4	5	6

L 数组：

	0	0	2	3	3	0

0	1	2	3	4	5	6

R 数组：

5	2	6	6	5	6	
0	1	2	3	4	5	6

i=1	L[2]=L[1]	R[0]=R[1]
i=2	L[5]=L[4]	R[3]=R[4]
i=3	L[6]=L[5]	R[3]=R[5]
i=4	L[6]=L[3]	R[2]=R[3]
i=5	L[6]=L[2]	R[0]=R[2]

NOIP2017

1.（快速幂）请完善下面的程序，该程序使用分治法求 x^p mod m 的值。（第一空 2 分，其余 3 分）

输入：三个不超过 10000 的正整数 x，p，m。

输出：x^p mod m 的值。

提示：若 p 为偶数，$x^p=(x^2)^{p/2}$；若 p 为奇数，$x^p=x*(x^2)^{(p-1)/2}$。

```
#include <iostream>
using namespace std;
int x, p, m, i, result;
int main() {
  cin >> x >> p >> m;
  result = ________(1)________ ;
  while ( ________(2)________ ) {
    if (p % 2 == 1)
    result = ________(3)________ ;
    p /= 2;
    x = ________(4)________ ;
  }
  cout << ________(5)________ << endl;
  return 0;
}
```

解答：

本题实现的是快速幂算法，首先第 ① 空中 result 需要赋初值 1。

因为 x^0 = 1。第 ② 空需要填循环继续的条件，显然每次循环 p 都会除以 2 下取整，每次通过判断 p 的最低位是否为 1 来修改 result，不难得出循环继续的标志是 p 大于 0，因为 p 为 0 之后最低为就不可能会是 1 了。故第 ① 空可以填 p>0，p!=0 或 p。

对于第 ③④ 空，考虑 p 的二进制表示，当第 i 位为 0 的时候，result 需要乘上 x^{2^i}，这个值可以通过每次循环 x = x * x % m 实现，故第 ④ 空需要填 x * x % m。当 p 最低位是 1 时，result 需要乘上前述的对应次幂，故第 ③ 空应该填上 result * x % m。

最后需要输出 result，第 ⑤ 空应该填上 result。

填空后的程序：

```
#include <bits/stdc++.h>
using namespace std;
//求 x 的 p 次方%m
int x, p, m, i, result;
int main() {
    cin >> x >> p >> m;
    //第 1 空：result = 1;
    result =  1;
    //第 2 空：p != 0 或者 p > 0
    while (p != 0) {
        if (p % 2 == 1)
            //第 3 空
            result = result * x % m;
        p /= 2;
        //第 4 空 x * x % m
        x = x * x % m;
    }
    //第 5 空：result
    cout << result<< endl;
    return 0;
}
```

2.（切割绳子）有 n 条绳子，每条绳子的长度已知且均为正整数。绳子可以以任意正整数长度切割，但不可以连接。现在要从这些绳子中切割出 m 条长度相同的绳段，求绳段的最大长度是多少。（第一、二空 2.5 分，其余 3 分）

输入：第一行是一个不超过 100 的正整数 n，第二行是 n 个不超过 10^6 的正整数，表示每条绳子的长度，第三行是一个不超过 10^8 的正整数 m。

输出：绳段的最大长度，若无法切割，输出 Failed。

```
#include <iostream>
using namespace std;
int n, m, i, lbound, ubound, mid, count;
int len[100]; // 绳子长度
int main() {
  cin >> n;
  count = 0;
  for (i = 0; i < n; i++) {
      cin >> len[i];
      ________(1)_______;
  }
  cin >> m;
  if ( ________(2)_______ ) {
      cout << "Failed" << endl;
      return 0;
  }
  lbound = 1;
  ubound = 1000000;
  while ( ________(3)_______ ) {
      mid = ________(4)_______ ;
```

```
        count = 0;
        for (i = 0; i < n; i++)
                (5)          ;
        if (count < m) ubound = mid - 1;
        else
        lbound = mid;
    }
    cout << lbound << endl; return 0;
}
```

解答：填空位置 ①：count = count + len[i]
填空位置 ②：count < m
填空位置 ③：lbound < ubound
填空位置 ④：(lbound + ubound + 1) / 2
填空位置 ⑤：count = count +len[i] / mid

题目解析
结合第 ①② 空可知，当绳段长度为 1 时可以切出的绳段个数最多，即所有绳子的长度相加，所以输出 Failed 的条件应该是所有绳子的总长度小于 mm。故用 count 计算总长度，第 ① 空填 count = count + len[i] 或 count += len[i]，第二空填 count < m 或 m < count。
第 ③④⑤ 空均在二分里，第 ③ 空是二分的结束条件，故应该填 lbound < ubound 或者 ubound < lbound，即当前二分的区间长度还大于 11 的时候，说明没有确定最终值是多少。第 ④ 空需要特别注意，由于底下 ubound 是通过 mid - 1，lbound 是通过 mid 来更新的，所以这空需要填 (lbound + ubound + 1) / 2 或 (lbound + ubound + 1) >> 1 或 (lbound + ubound) / 2 + 1，三者意思相同。
第 ⑤ 空是用来求当确定绳子的切割长度为 mid 后，最多能切多少段绳段，故应该填 count = count + len[i] / mid 或者 count += len[i] / mid。

填空后的程序如下：

```
#include using namespace std;
int n, m, i, lbound, ubound, mid, count;
int len[100]; // 绳子长度
int main() {
    cin >> n;
    count = 0;
    for (i = 0; i < n; i++) {
        cin >> len[i];
        //第 1 空：求总绳子的长度 count = count + len[i];
        count = count + len[i];
    }
    cin >> m;
    //第 2 空：如果总绳子长度<绳子根数，则不能切割
    //count < m
    if (count < m) {
        cout << "Failed" << endl;
        return 0;
    }
    //切割范围一定在 1~10^6 之间，因为每根绳子的长度在这个范围内
    lbound = 1;
    ubound = 1000000;
    //第 3 空：循环条件：lbound < ubound
```

```
    while (lbound < ubound) {
        //第 4 空：由于底下 ubound 是通过 mid - 1
        //lbound 是通过 mid 来更新的，所以这空需要填 (lbound + ubound + 1) / 2
        mid = (lbound + bound + 1) / 2;
        count = 0;
        for (i = 0; i < n; i++)
            //第 5 空：计算如果切割绳子长度为 mid，可以最多切多少根
            count = count +len[i] / mid;
        //如果绳子数量少了，说明 mid 太大，修改 ubound 的值
        if (count < m) ubound = mid - 1;
        else
            lbound = mid;
    }
    cout << lbound << endl;
    return 0;
}
```

NOIP2016

1. （读入整数）请完善下面的程序，使得程序能够读入两个 int 范围内的整数， 并将这两个整数分别输出，每行一个。（第一、五空 2.5 分，其余 3 分）输入的整数之间和前后只会出现空格或者回车。输入数据保证合法。

例如：

输入：

```
123   -789
```

输出：

```
123
-789
```

```
#include <iostream>
using namespace std;
int readint() {
        int num = 0;       // 存储读取到的整数
        int negative = 0; // 负数标识
        char c;            // 存储当前读取到的字符
        c = cin.get();
        while ((c < '0' || c > '9') && c != '-')
          c =   (1)   ;
        if (c == '-') negative = 1;
        else
          (2)   ;
        c = cin.get();
        while (   (3)   ) {
            (4)   ;
            c = cin.get();
        }
        if (negative == 1)
           (5)   ;
        return num;
}
  int main() {
     int a, b;
```

```
        a = readint();
        b = readint();
        cout << a << endl << b << endl;
        return 0;
    }
```

题目答案

填空位置 ①：cin.get()

填空位置 ②：num = c - '0'

填空位置 ③：c > '0' && c < '9'

填空位置 ④：num = num * 10 + c - '0'

填空位置 ⑤：num = -num

题目解析

通读题面和程序之后，我们可以发现 readint 函数将整数按字符一位一位读入，并还原成原数，它的处理流程如下：

如果开头是除了数字字符和负号之外的其他字符，我们不对其做任何处理，因此空 ① 应该填 cin.get()；

跳出第一个 while 循环代表我们读到了一个数字字符或者负号，如果读到的是负号，negative 赋值为 1，标识这个数为负数；否则就是读入了一个数字字符，我们需要利用 ASCII 码将 c 中存储的数字字符转换成它对应的数字，因此空 ② 处应该填 num = c - '0' 或 num = c - 48；

在第二个 while 循环中，我们应该将每一个读入的数字字符转换成对应的数字，因此空 ③ 应为 c > '0' && c < '9' 或 c > 48 && c < 57，保证了我们只处理数字字符，也就是处理到这个整数的结尾；

由于我们是从高位往低位读入的，因此每次需要给 num * 10 再加上新读入的数字。例如对于 12345，我们先读入 1，然后读入 2，此时的 num 为 1 * 10 + 2 = 12，接下来读入 3，此时 num 为 12 * 10 + 3 = 123，以此类推。因此空 ④ 应为 num = num * 10 + c - '0' 或 num = num * 10 + c - 48；

最后，如果负数标识 negative 为 1，表示 num 为负数，在此处我们可以将 num 变为其相反数，也可以直接返回 num 的相反数，这里答案很多。一般来说，空 ⑤ 会填 num = -num 或 num = -1 * num 或 return -num 或 return -1 * num。

填空后的程序：

```
#include <iostream>
using namespace std;
int readint() {
    int num = 0; // 存储读取到的整数
    int negative = 0; // 负数标识
    char c;
    c = cin.get(); // 存储当前读取到的字符

    while ((c < '0' || c > '9') && c != '-')
        //第 1 空：如果第一个字符 c 不是数字，也不是负号-，则重新读入
        c = cin.get();
    if (c == '-')
        negative = 1;
    else
       //第 2 空：如果 c 是数字，则存入 num
       num = c - '0';

    //上面是处理第一个读入的有效字符-------------
```

```
    c = cin.get();
    //第 3 空：c >= '0' && c <= '9'
    while (c >= '0' && c <= '9') {
        //第 4 空：将读入的字符转换连接到整数上
        num = num * 10 + c - '0';
        c=cin.get();
    }

    //如果是负数，将 num 修改为负
    if (negative == 1)
        //第 5 空：num = -1 * num
        num = -1 * num;
    return num;
}
int main() {
    int a, b;
    a = readint();
    b = readint();
    cout << a << endl << b << endl;
    return 0;
}
```

2. （郊游活动）有 n 名同学参加学校组织的郊游活动，已知学校给这 n 名同学 的郊游总经费为 A 元，与此同时第 i 位同学自己携带了 Mi 元。为了方便郊 游，活动地点提供 B(≥n)辆自行车供人租用，租用第 j 辆自行车的价格为 Cj 元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管 理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多 有多少位同学能够租用到自行车。（第四、五空 2.5 分，其余 3 分）

本题采用二分法。对于区间[l, r]，我们取中间点 mid 并判断租用到自行 车的人数能否达到 mid。判断的过程是利用贪心算法实现的。

```
#include <iostream>
using namespace std;
#define MAXN 1000000
    int n, B, A, M[MAXN], C[MAXN], l, r, ans, mid;
    bool check(int nn) {
      int count = 0, i, j;
      i =__(1)__;
      j =  1;
      while (i <= n) {
            if (___(2)___)
                count += C[j] - M[i];
            i++;
            j++;
      }
      return___(3)___;

    }

    void sort(int a[], int l, int r) {
        int i = l, j = r, x = a[(l + r) / 2], y;
```

```
        while (i <= j) {
            while (a[i] < x) i++;
            while (a[j] > x) j--;
            if (i <= j) {
               y = a[i]; a[i] = a[j]; a[j] = y; i++; j--;
            }
        }
        if (i < r) sort(a, i, r);
        if (l < j) sort(a, l, j);
    }

    int main() {
        int i;
        cin >> n >> B >> A;
       for (i = 1; i <= n; i++) cin >> M[i];
       for (i = 1; i <= B; i++) cin >> C[i];
        sort(M, 1, n);
        sort(C, 1, B);
        l = 0;
        r = n;
        while (l <= r) {
         mid = (l + r) / 2;
         if (     (4)     ) {
             ans = mid;
             l = mid + 1;
         } else
             r =      (5)      ;
    }
    cout << ans << endl;
    return 0;
}
```

题目答案

填空位置 ①：n - nn + 1

填空位置 ②：M[i] < C[j]

填空位置 ③：count <= A

填空位置 ④：check(mid)

填空位置 ⑤：mid - 1

题目解析

这类题目可以先从题面和主函数看起，把握程序的整体结构后，再去分析每一个子函数的实现方法。

主函数中先读入了数组 M 和 C 的每一个元素，分别代表每位同学携带的钱数和每辆自行车价格，随后使用自定义的 sort 函数连个数组进行排序。

sort 函数是典型的快排，会将数组 a 中，下标 l 到 r 的部分从小到大进行排序。

接下来使用二分法来寻找最多可以让多少人在总经费 A 元不花完的情况下租到车。因此，空 ④ 中需要填写 check(mid)；如果能进到 if 中，可以至少让 mid 人租到自行车，我们可以继续在 mid + 1 到 r 的范围内继续寻找；

否则，我们就需要在 l 到 mid - 1 这个范围内寻找有多少人可以租到车，因此空 ⑤ 应该填 mid - 1。

接下来我们回头去看 check 函数。它能判断是否能有 nn 人租到车，如果这 nn 人租车需要借的钱不大于总经费 A，就返回 true。

它采用了贪心算法，让钱多的人尽可能去租便宜的自行车。由于每个人的钱数和每辆车的租金已经从小到大排序过了，那么我们让第 n - nn + 1 人租第 1 辆车，让第 n - nn + 2 人租第 2 辆车… 第 n 个人租第 nn 辆车。让钱数最多的 nn 人依次去租最便宜的 nn 辆车，因此空 ① 应该填 n - nn + 1。

如果一个人的钱数 M_i 不够租价格为 C_j 的自行车，就需要从总经费中借 C[j] - M[i]，因此空 ② 应该是 M[i] < C[j] 或 M[i] <= C[j]。

count 中存储了一共需要借多少经费，因此我们需要在空 ③ 处写 count <= A，如果需要借的经费不大于总经费 A 元，表示可以让钱数最多的 nn 人租到自行车。

NOIP2015

1.(打印月历）输入月份m（1 ≤m ≤ 12)，按一定格式打印2015 年第m 月的月历。(第 三、四空2.5 分， 其余3 分）

例如，2015 年1 月的月历打印效果如下（第一列为周日）。

```
S   M   T   W   T   F   S
                1   2   3
4   5   6   7   8   9   10
11  12  13  14  15  16  17
18  19  20  21  22  23  24
25  26  27  28  29  30  31
```

```
#include <iostream>
using namespace std;
const int dayNum[]= {-1, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31};
int m,offset, i;
int main() {
    cin >> m;
    cout << " S\tM\tT\tW\tT\tF\tS "<< endl; // '\t'为 TAB 制表符
    ___(1)___;
    for (i = 1; i < m; i++)
        offset = ____(2)____;
    for (i = 0; i < offset; i++)
        cout <<'\t';
    for (i = 1; i <= ___(3)___;i++)
    {
```

```
        cout << (4)  ;
        if (i == dayNum[m] ||     (5)    == 0)
            cout << endl;
        else
         cout << '\t';
     }
    return 0;
}
```

题目答案

填空位置 ①：offset = 4

填空位置 ②：(offset + dayNum[i]) % 7

填空位置 ③：dayNum[m]

填空位置 ④：i

填空位置 ⑤： (offset + i) % 7

题目解析

通读题面和程序之后，我们发现这个程序的功能是按照格式打印出输入月份 mm 的月历。

不难发现 offset 的初值不会是全局变量的默认初值 00，所以 ① ② 两空都跟 offset 有关。观察后面的程序可以得知，offset 变量的作用是在后面决定日历日期第一行前面要空几个制表符，因此 ① 应该是对 offset 赋初值，即一月份日期第一行前面需要空几个制表符，由例子图片可以知道需要空 4 个，故空 ① 应该填 offset = 4。

② 空则跟计算第 2 到 12 月份 offset 值有关，因为一星期有 7 天，所以下个月份的 offset 和上个月份 offset 的关系应该是 (offset+dayNum[i]) % 7，dayNum[i] 是第 ii 个月的天数，故 ② 空应该填 (offset+dayNum[i]) % 7。

之后的程序就是打印月历的每一天，因此循环的次数应该是这个月 m 的天数，故 ③ 空应该填 dayNum[m]。循环遍历 i 表示当前打印第几天，所以 ④ 空应该填 i。可以看出第 ⑤ 空是打印月历换行的条件，第一个条件是日期打印到了最后一天，第二个条件自然是当前位置是不是该行的第 7 个位置，这个可以用 offset + i 是否为 7 的倍数来判断，故第 ⑤ 空应该填 (offset+i) % 7。

2. (中位数)给定n (n 为奇数且小于1000)个整数，整数的范围在0~m (0 < m < 2^{31}) 之间，请使用二分法求这n 个整数的中位数。所谓中位数，是指将这n 个数排序之后，排在正中间的数。(第五空2 分，其余3 分)

```
#include <iostream>
using namespace std;
const int MAXN = 1000;
int n, i, lbound, rbound, mid, m, count;
int x[MAXN];
int main(){
    cin >> n >> m;
    for (i = 0; i < n; ++i)
        cin    >> x[i];
        lbound=0;
        rbound=m;
```

```
    while (    (1)    ) {
        mid  = (lbound + rbound )/2;
                (2)      ;
        for (i = 0; i < n; i++)
            if (    (3)    )
                (4)      ;
        if (count > n / 2) lbound = mid + 1;
        else
            (5)      ;
    }
    cout << rbound << endl;
    return 0;
}
```

题目答案

填空位置 ①：lbound < rbound

填空位置 ②：count = 0

填空位置 ③：x[i] > mid

填空位置 ④：count++

填空位置 ⑤：rbound = mid

题目解析

这题代码填空主要考察二分的思想，由于比中位数大的数不会超过 n/2，所以可以二分最后中位数的值，当比该值的数大的数超过了 n/2，说明该值太小了，反之说明该值太大了，二分结束时即可求出中位数的值是多少。

第 ① 空需要填的是二分继续的判断条件，即当二分的区间长度大于 1 的时候，还需要继续二分，故第 ① 空需要填 lbound < rbound 或者 rbound > lbound。

观察可以发现二分法里题目给出的代码中，有对 count 进行 +1 的操作，但是没有重新赋初值，故第 ② 空应该是将 count 重新初始化为 0，即 count = 0。

第 ③ ④ 空需要填写修改 count 变量的操作，从后面对二分区间的改变可以看出，当 count 太大的时候，需要增大左端点，因为 count 的含义应该是比当前值 mid 大的数的个数，所以第 ③ 空应该填 x[i]>mid 或 mid<x[i]，第 ④ 空应该填 count++ 或者 count = count + 1。

第 ⑤ 空条件语句的上一个分支是修改 lbound 的值，故第 ⑤ 空应该是修改 rbound，即右端点的值，所以第 ⑤ 空应该填 rbound = mid。