CSP-J 第一轮学军中学信友队开放模拟赛

一、判断题（每题 1 分，共 10 分，选项 T=正确，F=错误，每空标号处均填选择的答案对应的大写字母）

1. CSP-J/S 非专业级别比专业级别更难（ F ）

题目解析：非专业级别较之专业级别更为简单

2. CSP-J/S 非专业认证在校生与在职人员皆可参加（ T ）

题目解析：CSP 非专业认证不设年龄、性别限制，在校生和在职人员均可参加

3. CCF 建议将 CSP-J/S 成绩作为职业晋升和升学的唯一依据（ F ）

题目解析：不建议将 CSP 成绩作为职业晋升和升学的唯一依据，不建议以功利的心态参加 CSP 认证。

4. CSP-J 和 CSP-S 皆举办两轮（ T ）

题目解析：CSP-J 和 CSP-S 均分别举办两轮：CSP-J1，CSP-S1 及 CSP-J2、CSP-S2

5. 可以直接参加 CSP-J/S 第二轮（ F ）

题目解析：参加 CSP-J/S 第二轮，必须先参加第一轮，达到一定的分数者方可参加第二轮

6. CSP-J/S 受理因选手个人失误提交的申诉（ F ）

题目解析：选手个人原因，CCF 都不得理你

7. CSP-J/S 属于 NOI 系列赛事（ T ）

题目解析:CSP-J/S 认证成绩优异者,可参加 NOI 省级选拔,省级选拔成绩优异者可参加 NOI。因此属于 NOI 系列赛事

8. 可以在 CSP-J/S 赛前发布虚假的“CSP-J/S”试题解析”干扰认证秩序（ F ）

9. CSP-J/S 允许选手自带鼠标/键盘（ F ）

题目解析：初赛只允许带支笔，复赛只允许带上你的人

10. CSP-J/S 认证者可在认证完成离开考场后返回考场（ F ）

题目解析：一锤子买卖，交卷后你就应该各回各家

以上所有题目都属于政策题目，关于 CSP 的规定如下：

2014 年，CCF 推出 CSP 认证（Certified Software Professional，软件能力认证），以评价计算机专业人士或准专业人士计算机科学的基础能力----算法和编程能力。CSP 每年举办 3 次，现已成为一些企业及许多大学评价计算机专业大学生专业能力的重要工具。现 CCF 推出 CSP 非专业级别的能力认证。

1. 非专业级别较之专业级别更为简单，将分两个级别进行，分别为 CSP-J（入门级，Junior）和 CSP-S（提高级，Senior），均涉及算法和编程。

2. CSP-J 和 CSP-S 均分别举办两轮：CSP-J1，CSP-S1 及 CSP-J2、CSP-S2，认证方式均为现场认证，非网络认证。参加 CSP-J/S 第二轮，必须先参加第一轮，达到一定的分数者方可参加第二轮。

3. CCF 在各省设立认证组织单位，授权其安排认证点、监督认证过程以及处理与认证相关的事务。

4. CSP 非专业认证不设年龄、性别限制，在校生和在职人员均可参加，但必须满足下述两个条件。

5. 参加认证者必须事先在网上注册并完成相关报名手续，主办单位审核通过后方可参加认证。

6. 参加认证者必须选择一个认证点并得到认证组织单位的同意。

本学会再次重申,CSP 是符合本学会定位及符合学会章程的一项专业能力评价活动,而 CSP-J 和 CSP-S 是面向有兴趣者自愿的一项计算机科学活动，旨在推动计算机科学的普及，让更多的青少年和非专业人士接触和学习计算机科学，并对他们未来选择以计算机为其职业有所帮助。CSP 认证不纳入行政轨道，不建议将 CSP 成绩作为职业晋升和升学的唯一依据，不建议以功利的心态参加 CSP 认证。

二、单选题（每题 2 分，共 30 分）

1. 以下编程语言中，同时符合面向对象与编译执行的是（ B ）

A. C　　B. C++　　C. Pascal　　D. Python

题目解析：4 个选项中面向对象的只有 C++和 Python。因为 Python 是脚本语言，并且脚本语言都是解释执行的，所以答案选 B。另外还有

(1) JavaScript 是面向对象的脚本语言

(2) bat、perl 是面向过程的脚本语言

(3) C#、Java 是面向对象的非脚本语言

2. 二进制补码求和：11010010+00000101=(C)

A. 11000011　　B. 10110110　　C. 11010111　　D. 10001001

题目解析：按 2 进制相加，逢 2 进 1

3. 与十进制数 17.5625 相对应的 8 进制数是（ B ）

A. 21.5625　　B. 21.44　　C. 21.73　　D. 21.731

题目解析：$2 \times 8^{1} + 1 \times 8^{0} + 4 \times 8^{-1} + 4 \times 8^{-2}=17.5625$

4. 下列网络设备中，不属于局域网设备的是（ 无正确选项 ）

A. 路由器　　B. 集线器　　C. 网卡　　D. 中继器

题目解析：以上设备都是局域网设备。参考答案选 A，不清楚出题人的脑回路

5. 以下不属于结构化程序的结构是（ B ）

A. 顺序结构　　B. 输入输出结构　C. 分支结构　　D. 循环结构

题目解析：程序的基本结果有且仅有 3 种：顺序、分支、循环

6. 在待排序文件已基本有序的前提下，下述排序方法中效率最高的是（ A ）

A. 插入排序　　B. 选择排序　　C. 快速排序　　D. 归并排序

题目解析：

(1) 若序列基本有序，插入排序的比较次数会接近 1 次，时间复杂度为 O(N)

(2) 若序列基本有序，快速排序的时间复杂度会由 O(NlogN)退化成 $O(N^2)$

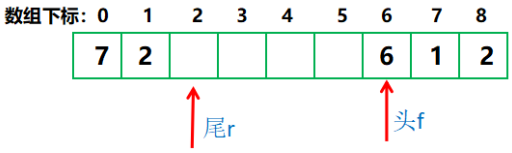
(3) 选择排序和冒泡排序，在任何情况下，时间复杂度都是 $O(N^2)$

(4) 归并排序核堆排序，在任何情况下时间复杂度都是 O(NlogN)

7.设循环队列中数组的下标范围是 0～m-1，其头尾指针分别为 f 和 r，则其元素个数为(D)

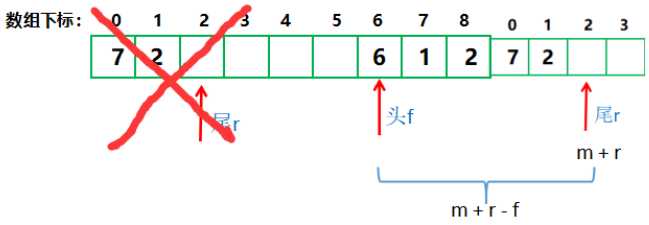
A. r-f　　B. r-f+1　　C. (r-f+1)%m　　D. (r-f+m)%m

题目解析：首先 A、B 肯定是不正确的，因为循环队列的 f、r 会出现下面的情况



求解方法 1：画几个如上所示的队列，将 r、f 的值带入到 C、D 选项中进行运算

求解方法 2：假设将前面的 r 个空间连接到数组的后面，则整个数组的长度就变成 m+r



此时数组的元素的个数为 m + r − f ，另外当队列为空时有 r == f，因此数组中的元素个数为 (m + r − f) % m

8. 现有 A、B、C、D、E、F、G 七个元素依次进栈操作，但可随时出栈，出了栈后不能再进栈，下面哪个是不可能的(B)

A. ABCDEFG　　B. ADBCGFE　　C. ABCDEGF　　D. GFEDCBA

题目解析：一个个选项排序，选项 B 中 C 必须在 B 前面出栈

9. 关于拓扑排序，下面说法正确的是（ D ）

A. 所有连同的有向图都可以实现拓扑排序

B. 对一个图而言，拓扑排序的结果是唯一的

C. 拓扑排序中入度为 0 的结点总会排在入度大于 0 的结点的前面

D. 拓扑排序结果序列中的第一个结点一定是入度为 0 的点

题目解析：拓扑排序的规则为：

(1). 找到一个入度为 0 的顶点，将其添加到队列中。

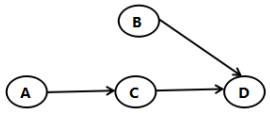
(2). 将当前队首的元素出队列，并输出该顶点，将与其相邻的顶点的入度减 1，若顶点的入度变为 0，则将该顶点添加到队列。

(3). 重复步骤(2)，直到访问完所有的顶点为止

选项 A，要实现拓扑排序，必须得有入度为 0 的顶点，所以错误

选项 B，若有多个入度为 0 的顶点，顶点入队列的顺序不同，最后的结果也不同，所以错误

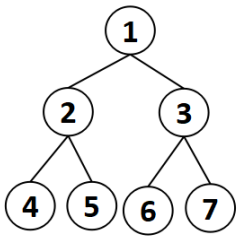
选项 C，前后顺序只与进出队列的先后有关，对于下图 ACBD 和 ABCD 都是正确的拓扑序列



10. 完全二叉树的顺序存储方案，是指将完全二叉树的结点从上至下，从左到右依次存放到一个顺序结构的数组中。假定根结点存放在数组的 1 号位置，则第 k 号结点的父结点如果存在的话，应当存放在数组的(C)号位置。

A. 2k　　B. 2k+1　　C. k/2 下取整　　D. (k+1)/2 下取整

题目接下：如下图，节点 i 的左孩子的下标为 2*i，右孩子的下标为 2*i + 1，因此每个节点的父节点所在的位置为 k/2 下取整。



11. 以下逻辑表达式的值恒为真的是（ A ）。

A. P∨(¬P∧Q)∨(¬P∧¬Q)　　B. Q∨(¬P∧Q)∨(P∧¬Q)

C. P∨Q∨(P∧¬Q)∨(¬P∧Q)　　D. P∨¬Q∨(P∧¬Q)∨(¬P∧¬Q)

题目解析：P、Q 分别取真、假来进行推导

12. 表达式（1+34）*5-56/7 的后缀表达式为（ C ）

A. 1+34*5-56/7　　B. -*+1 34 5/56 7

C. 1 34+5*56 7/-　　D. 1 34 5*+56 7

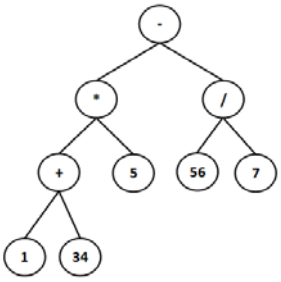
E. 1 34+5 56 7-*/

题目解析：将表达式画成如下的二叉树后，后序遍历所得的序列即为后缀表达式。

先序遍历：“根左右”，也称先根遍历。先遍历根节点，再遍历左子树，最后遍历右子树

中序遍历：“左根右”，也称中根遍历。先遍历左子树，再遍历根节点，最后遍历右子树

后序遍历：“左右根”，也称后根遍历。先遍历左子树，再遍历右子树，最后遍历根节点



13. 某大学计算机专业的必修课及其先修课程如下表所示：

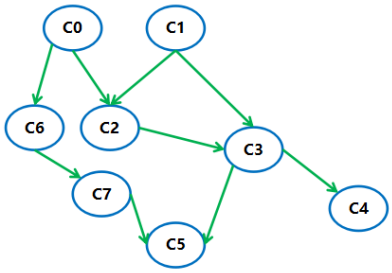
课程代号	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7
课程名称	高等数学	程序设计语言	离散数学	数据结构	编译技术	操作系统	普通物理	计算机原理
先修课程			C_0, C_1	C_1, C_2	C_3	C_3, C_7	C_0	C_6

下列课程安排方案哪个不是合理的（ D ）。

A. C0, C1, C2, C3, C6, C7, C5, C4　　B. C0, C1, C2, C3, C4, C6, C7, C5

C. C0, C1, C6, C7, C2, C3, C4, C5　　D. C0, C1, C6, C7, C5, C2, C3, C4

题目解析：按课程学习顺序，汇制如下有向图，不合理的拓扑序列即为不合理的安排方案。



选项 D，学习 C5 前必须要学习 C3，因此不合理

14. 在 n 个结点的顺序表中，算法的时间复杂度是 O(1)的操作是（ A ）。

A．访问第 i 个结点（1≤i≤n）和求第 i 个结点的直接前驱（2≤i≤n）

B．在第 i 个结点后插入一个新结点（1≤i≤n）

C．删除第 i 个结点（1≤i≤n）

D．将 n 个结点从小到大排序

题目解析：

选项 B 的时间复杂度为 O(n)，因为要将 i 后面的所有节点，往后挪一个位置

选项 C 的时间复杂度为 O(n)，因为要将 i 后面的所有节点，往前挪一个位置

选项 D 的时间复杂度为 $O(n^2)$ 或者 O(nlogn)，取决于所采用的排序方法

15.给定一个正整数 N=8934632178，现决定依次删除其中 6 个数位上的数字（每次删除一个数位上的数字），每次删除后按原来的次序组成一个新数 M 的值均是当前状态下的最小数，则第四次应该删除的数字是（ D ）。

A、6　　B、8　　C、7　　D、4

题目解析：没有想到比较好的方法，只能一个个枚举，每次都找字典序最小的

三、问题解答（每题 5 分，共 10 分，每空标号处均填选择的答案对应的大写字母）

1. 将 2 个红球，1 个蓝球，1 个白球放到 10 个编号不同的盒子中去，每个盒子最多放一个球。有多少种放法(B)

A. 5040　　B. 2520　　C. 420　　D. 1260

题目解析：假设两个红球不一样，则最后的结果为：10 x 9 x 8 x 7 = 5040。

接下来考虑 2 个红球相同的情况。

先放蓝球，则有 10 种选择；再放白球则有 9 种选择；最后放 2 个红球有 C(8, 2)种选择。

因此方案总数为：10 x 9 x C(8, 2) = 10 x 9 x (8x7)/2 = 2520

2.两个人抛硬币，规定第一个抛出正面的人可以吃到苹果，请问先抛的人能吃到苹果的概率多大（ B ）

A. 1/2　　B. 2/3　　C. 3/4　　D. 5/8

题目解析：

若每人只能抛一次，则概率为 1/2；

若每个人可以抛无限次，则概率为 2/3。因为先抛的人吃到苹果的概率：$1/2 + 1/2^3 + 1/2^5 + 1/2^7 \ldots$ 等比数列求和后求极限，可得结果为 2/3

$$\lim_{n\to\infty}\left(\frac{2-\frac{2}{4^n}}{3}\right)=\frac{2-0}{3}=\frac{2}{3}$$

四、阅读程序（每题 6 分，共 24 分，每空标号处均填选择的答案对应的大写字母）

1.

```
#include <cstdio>
int main()
{
    int i,s,max;
    int a[11];
    for(i=1;i<=10;i++)
        scanf("%d",&a[i]);
    max=a[1];
    s=a[1];
    for(i=2;i<=10;i++){
        if(s<0)      // 题目解析：如果前面的累加和为 0，则将 s 赋为 0，
            s=0;     // 这是求最大连续子段和的典型代码
        s+=a[i];
        if(s>max)
            max=s;
    }
    printf("max=%d\n",max);
    return 0;
}
```

输入：8 9 -1 24 6 5 11 15 -28 9

输出：max=(B)

A. 24　　B. 77　　C. 70　　D. 61

题目解析：程序功能，求最大连续子段和

	1	2	3	4	5	6	7	8	9	10	
数组 a:	8	9	-1	24	6	5	11	15	-28	9	
s的值:	8	17	16	40	46	51	62	77	49	58	

s 的最大值为 77。max=s，因此答案选 B

2.

```
#include<iostream>
#include<string>
using namespace std;
int main()
{
    string map= "22233344455566677778889999";
    string tel;
    int i;
    cin>>tel;
    for(i=0;i<tel.length();i++)
       if((tel[i]>='0') && (tel[i]<='9') )
           cout<<tel[i];
       else if( (tel[i]>='A') && (tel[i]<='Z'))
           cout<<map[tel[i]-'A'];
    cout<<endl;
    return 0;
}
```

输入:CCF-CSP/J-2019

输出: D

A. 223-287/5-2019 B. 2262181592019 C. 226-285/9-2019 D. 22328752019

题目解析:将字符串中的数字字符原样输出;大写字母换成 map 中对应位置的数字字符输出;其他字符不输出。**(特别提醒:本题不需要知道字符具体的 ASCII 码值,‘A’‘B’‘C’‘D’从 map 的第 1 个往后数就行了)**,因此正确选项为:D

CCF-CSP/J-2019

223 287 5 2019

3.

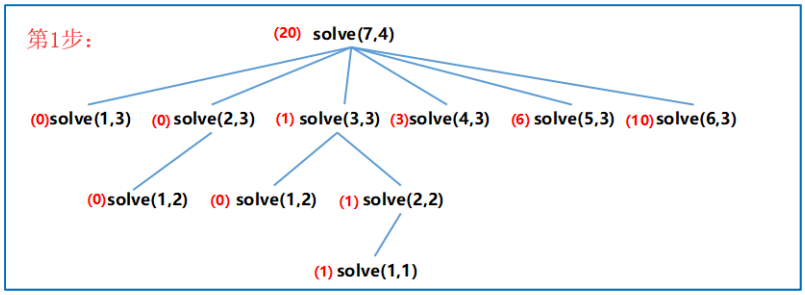
```
#include<iostream>
using namespace std;
int solve(int n,int m) {
    int i,sum;
    if(m==1) return 1;
    sum=0;
    for(i=1;i<n;i++)
        sum+= solve(i,m-1);
    return sum;
}
int main()
{
    int n,m;
    cin>>n>>m;
    cout<<solve(n,m)<<endl;
    return 0;
}
```

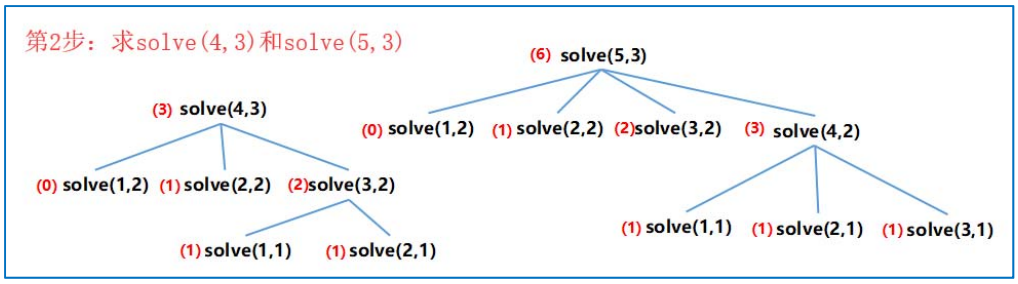
输入：7 4

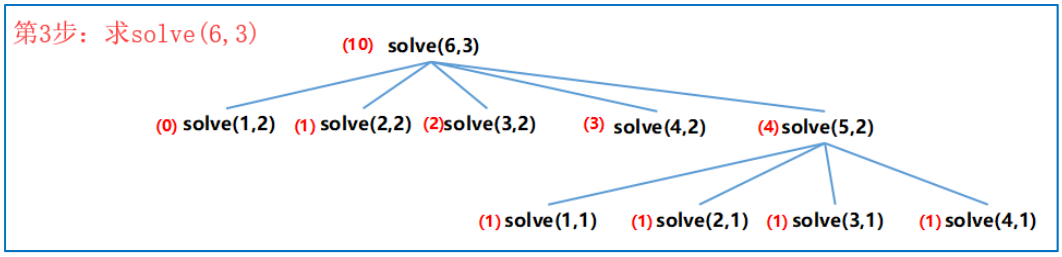
输出： A

A. 20 B. 28 C.21 D.24

题目解析：画递归推导图







4.

```
#include <stdio.h>
int main()
{
    int i,j,k;
    int a[101];
    for(i=0;i<=100;i++)
        a[i]=i;
    for(k=5;k>=2;k--){
        for(i=1;i<=100;i++) // 第 1 步：若数组下标是 k 的倍数，则删除该元素
            if(i%k==0)
                a[i]=0;
        for(i=1;i<=99;i++)  // 第 2 步：利用冒泡排序，将 a 由小到大排序
            for(j=1;j<=100-i;j++)
                if(a[j]>a[j+1]){
                    a[j]=a[j]+a[j+1];
                    a[j+1]=a[j]-a[j+1];
                    a[j]=a[j]-a[j+1];
                }
    }
```

```
    j=1;
    while(a[j]==0&&j<100)
        j++;
    for(i=j;i<=100;i++) // 第 3 步：求数组中，所有非 0 元素的和
        a[0]=a[0]+a[i];
    printf("%d\n",a[0]);
    return 0;
}
```

输出：C

A. 935　　B.235　　C.970　　D.870

题目解析：阅读本题代码时需要特别注意如下两点。

(1) 如果 if 和 for 语句没有大括号，则只会管辖到离它最近的第 1 个语句块

(2) 题目中的冒泡排序代码块包含在 for(k=5;k>=2;k--)中

本题的程序功能描述如下：

(1) 将数组 a 的初值设置为 0 - 100

(2) 若数组下标 i 是 5 的倍数，则 a[i]=0，然后将数组 a 中的所有元素按由小到大排序

(3) 若数组下标 i 是 4 的倍数，则 a[i]=0，然后将数组 a 中的所有元素按由小到大排序

(4) 若数组下标 i 是 3 的倍数，则 a[i]=0，然后将数组 a 中的所有元素按由小到大排序

(5) 若数组下标 i 是 2 的倍数，则 a[i]=0，然后将数组 a 中的所有元素按由小到大排序

(6) 忽略掉数组 a 中所有取值为 0 的元素，将数组中所有非 0 元素累加道 a[0]中，即：计算数组中所有非 0 元素的和

五、完善程序（前 5 空 2 分，后 4 空 4 分，共 26 分，每空标号处均填选择的答案对应的大写字母）

1. 高精度除法：

题目解析：本题求解的是两个高精度整数相除的商和余数。

求解思路：循环从被除数中减去除数，每减一次则本次的商增加 1。

```
#include<iostream>
#include<string>
#include<algorithm>
using namespace std;
const int MAXN=5005;
//Ans 是商 C 是余数
int A[MAXN], B[MAXN], Ans[MAXN] , C[MAXN], Len_A,Len_B,Len_Ans; void
Read(int *A) {
    string cur;
    cin>>cur;
    // 题目解析：用 A[0]保存高精度整数的长度。cur.length()可获取字符串的长度
    A[0]=cur.length();
// 题目解析:将数字字符转换成整数数字;‘0’的 ASCII 码为 48,也可写成 cur[i-1]- ‘0’
    for(int i=1; i<=A[0]; i++) A[i]=cur[i-1]-48;
    // 题目解析： reverse 是 STL 中的函数，其功能为将数组 A 中除 A[0]以外的元素反转
    // 即：假设数组 A 中的元素为 1 2 3，本行代码的功能是将数组 A 由 1 2 3 变成 3 2 1
    reverse(A+1,A+A[0]+1);
}
```

```
bool Big()
{ //比大小
    if(C[0]>B[0]) return true;
    if(C[0]<B[0]) return false;
    // 题目解析：若C数组和B数组的长度相同，则循环判断数组中的每一个整数
    for(int i=C[0]; i>=1; i--)
        if(C[i]<B[i]) return false;
        else if(C[i]>B[i]) return true;
    // 题目解析:因为要实现循环相减,所以可以确定Big()的作用是判断C>=B是否成立。
    // 所以这里应该是选项为A，true
    return   (1)    true ;
}

void Minus() { //减法
    int c=0;
    for(int i=1; i<=C[0]; i++) {
        // 题目解析:使用C中的值减去B中的值，高精度减法必然需要注意借位的情况。
        // 所以可以确定本空必然与B[i]和c有关。
        // 代码的思路逻辑为：若C[i] < 0，则表示需要向前借位并将c设置为1。
        // 因此本题的答案为选项D，B[i]+c。C[i]不仅要减B[i]还要减掉上一次的借位
        C[i]-=  (2)  B[i] + c;
        if(C[i]<0) {
            C[i]+=10;
            c=1;
        } else c=0;
    }
    while(C[0]>1&&C[C[0]]==0) C[0]--;  //去掉首位的0
}
void output(int *A) {
    for(int j=A[0]; j>=1; j--) cout<<A[j];
    cout<<endl;
}
```

```
int main() {
    Read(A);//被除数
    Read(B);//除数
    C[0]=0;
    for(int i=A[0]; i>=1; i--) {
        // 题目解析：将被除数 A 中下一个要参与运算的数添加到高精度数 C 中
        // 因此这里需要将 C 中原来的数都往后挪一个位置，所以本空为选项 F
        // C[j+1] = C[j]
        for(int j=C[0]; j>=1; j--)   (3)  C[j+1]=C[j];
        C[1]=A[i];
        // 题目解析：既然数组 C 中多了一个数，所以这里必然是选项 I，  C[0]++
          (4)  C[0] ++;
        while(C[0]>1&&C[C[0]]==0) C[0]--;
        while(Big()) {
            Minus();
        // 题目解析：题目注释中有说明，Ans 为商。
        // 所以每从高精度整数 C 中减掉一次高精度整数 B 后
        // 都需要将 Ans 对应位置的值增加 1
        // 因此本空应为选项 K，Ans[i]++
              (5)  Ans[i]++;
        }
    }
    Ans[0]=A[0];
    while(Ans[0]>1&&Ans[Ans[0]]==0) Ans[0]--;
    output(Ans);
    output(C);
    return 0;
}
```

1-5 空依次应该填：（A）（D）（ F）（ I）（ K）

A.true　　B. false　　C. B[i]　　D.B[i] +c　　E. c[j]=c[j+1].　　F. C[j+1]=C[j]

G. C[j-1]=c[j]　　H. c[j]=C[j-1]　　I. C[0]++　　J. C[0]--　　K. Ans[i]++　　L. Ans[0]--

2.

地鼠游戏

地鼠游戏是一项需要反应速度和敏捷判断力的游戏。游戏开始时，**会在地板上一下子冒出很多地鼠来，然后等你用榔头去敲击这些地鼠**，每个地鼠被敲击后，将会增加相应的游戏分值。问题是这些地鼠不会傻傻地等你去敲击，它总会在冒出一会时间后又钻到地板下面去（而且再也不上来），每个地鼠冒出后停留的时间可能是不同的，而且每个地鼠被敲击后增加的游戏分值也可能是不同，为了胜出，游戏参与者就必须根据每个地鼠的特性，有选择地尽快敲击一些地鼠，使得总的得分最大。

这个极具挑战性的游戏王钢特别喜欢，最近他经常在星期天上午玩这个游戏，慢慢地他不但敲击速度越来越快（敲击每个地鼠所需要的耗时是 1 秒），而且他还发现了游戏的一些特征，那就是每次游戏重新开始后，某个地鼠冒出来后停留的时间都是固定的，而且他记录了每个地鼠被敲击后将会增加的分值。于是，他在每次游戏开始后总能有次序地选择敲击不同的地鼠，保证每次得到最大的总分值。求最大游戏总分值。

题目解析：贪心算法。本题还可以用二叉堆来求解，题库中拥有同类题目（MZ1552）

(1) 如果两个地鼠停留时间相同，优先舍弃分数低的；

(2) 在满足(1)的情况下尽量敲击更多的地鼠。

从选项来看：

(1) 第（8）空只与地鼠的停留时间有关，并且在(8)、(9)空后会执行
a[k-1].time --，表示本轮未选择敲击该地鼠。且最后一个循环统计得分时，是以
a[i].time＞0 作为依据判断是否选择敲击该地鼠。因此可以确定第(8)、(9)空实现的
功能为：若停留时间相同则优先舍弃得分较低的地鼠。由此可以确定第(8)空为
a[k].time==a[k-1].time 选项 D。第（9）空为 a[k].v<=a[k-1].v 选项 A

(2) 同理，因为地鼠停留的时间为整数，所以
若 a[i-1].time!=a[i].time，则两只地鼠可以兼得；
若 a[i-1].time==a[i].time，则需舍弃地鼠。因此第(7)空为 a[i-1].time==a[i].time

(3) 如上步骤，可保证优先舍弃分数低的地鼠；根据贪心规则（1）求解过程中，我们是通过 a[i].time 来控制地鼠是否被舍弃，因此还需要保证交换到前面的价值更低的老鼠能尽量多停留一点儿时间。所以还需将地鼠按停留时间由小到大排序。所以第（6）空为 x.time<y.time，选项 A

```
#include <bits/stdc++.h>
using namespace std;
int n,ans;
struct node {
    int time,v;
} a[2000];
int cmp(node x,node y) {
    return   (6)  x.time<y.time 选项 A;
}
int main() {
    scanf("%d",&n);
    for (int i=1; i<=n; i++) scanf("%d",&a[i].time);
    for (int i=1; i<=n; i++) scanf("%d",&a[i].v);
    sort(a+1,a+n+1,cmp);
    for (int i=1; i<=n; i++) {
        if ( (7) a[i-1].time==a[i].time 选项 C ) {
            int k=i;
            while ( (8) a[k].time==a[k-1].time 选项 D &&k>1)
            {
                if ( (9) a[k].v<=a[k-1].v 选项 A ) swap(a[k],a[k-1]);
                a[k-1].time--;
                k--;
            }
        }
    }
    for (int i=1; i<=n; i++) if(a[i].time>0) ans+=a[i].v;
    printf("%d\n",ans);
}
```

(6). A. x.time<y.time B.x.time>y.time C.x.v<y.v D.x.v>y.v

(7). A. a[i].time==a[i+1].time B. a[i].v==a[i+1].v C a[i-1].time==a[i].time D.a[i-1].v==a[i].v

(8). A. a[k].time!=a[k-1].time B. a[k].time>a[k-1].time

C. a[k].time<a[k-1].time D. a[k].time==a[k-1].time

(9). A. a[k].v<=a[k-1].v B. a[k].v>=a[k-1].v

C. a[k].time<a[k-1].time D. a[k].time>a[k-1].time

参考答案：

一、判断题：1、F 2、T 3、F 4、T 5、F 6、F 7、T 8、F 9、F 10、F

二、单选题：1、B 2、C 3、B 4、A 5、B 6、A 7、D 8、B 9、D 10、C

11、A 12、C 13、D 14、A 15、D

三、数学题：1、B 2、B

四、阅读程序：1、B 2、D 3、A 4、C

五、完善程序：(1)、A (2)、D (3)、F (4)、I (5)、K (6)、A (7)、C (8)、D (9)、A