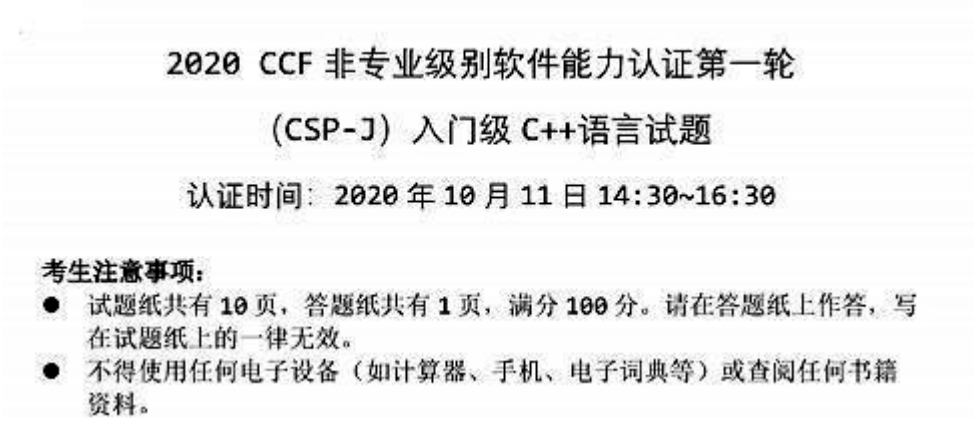
CCF CSP 认证（CCF 计算机软件能力认证 Certified Software Professional**)**

中国计算机学会（CCF）联合华为、360、滴滴等十余家知名 IT 企业以及清华、北航、国防科大等 15 所著名高校于 2014 年推出 CCF CSP（计算机软件能力）认证标准，用于评价业界人士的计算机软件能力

CSP-J ---- NOIP 普及组（初赛 、复赛）

CSP-S ---- NOIP 提高组（初赛、复赛）



2020 CCF 非专业级别软件能力认证第一轮

（CSP-J）入门级 C++语言试题

认证时间：2020 年 10 月 11 日 14:30~16:30

考生注意事项：

- 试题纸共有 **10** 页，答题纸共有 **1** 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。

分数组成：单项选择题 15 题 共：30 分

阅读程序题：3 题（判断、选择） 共 40 分

完善程序题：2 题 （选择） 共 30 分

目录

一、单项选择题

1. 存储单元 答案 B

在内存储器中每个存储单元都被赋予一个唯一的序号，称为（ ）。

A. 下标 B.地址 C. 序号 D. 编号

内存储器有多个存储单元组成，存储单元有编号，这些编号称为存储单元的地址号。

我们可以把内存想像成一连串带有编号的空房间，这个编号就是内存的地址。

1000	1001	1002	1003	1004	1005	1006	1007	.....

2. 编译器 答案 A

编译器的主要功能是()。

A.将源程序 翻译成机器指令代码 B.将一种高级语言 翻译成另一一种高级语言

C.将源程序重新组合 D.将低级语 言翻译成高级语言

编译器就是将“一种语言（通常为高级语言）”翻译为“另一种语言（通常为低级语言）”的程序。一个现代编译器的主要工作流程：源代码 (source code) → 预处理器(preprocessor) → 编译器(compiler) → 目标代码 (object code) → 链接器(Linker) → 可执行程序 (executables)。

3. 逻辑运算 答案 C

设 x = true，y = true，z = false，以下逻辑运算表达式值为真的是（ ）。

A. (x∧y)∧z B. x∧(z ∨ y)∧z C).(x∧y) ∨ (z ∨ x) D.(y∨z)∧x∧z

注：∧ 合取 表示 And 两个都为真是结果才为真。

逻辑”与”运算 &&（并且，and)

两种情况：只有两种情况都为真时，结果才为真。

A	B	结果
真	真	真
真	假	假
假	真	假
假	假	假

∨ 析取 表示 Or 两个有一个为真，结果为真

逻辑”或”运算 ||（或者，or）

A	B	结果
真	真	真
真	假	真
假	真	真
假	假	假

补充：¬ 非 ！ not

4. 信息存储单位 答案 B

现有一张分辨率为 2048×1024 像素的 32 位真彩色图像。请问要存储这张图像，需要多大的存储空间？ （ ）

A. 4MB B.8MB C.32MB D.16MB

字节（Byte ） = 8 bit(位) 32/8 =4

一个像素是 32 位真彩色，也就一个像素占 4 个字节

1M= 1024*1024B=2^20B

2048*1024*4 = 2*1024*1024*4=8*2^20 = 8M

2048 x 1024 x 4 / 1024 /1024 = 8M

一个英文字母（不分大小写）占一个字节的空间，一个中文汉字占两个字节的空间。英文标点占一个字节，中文标点占两个字节 [6] 。

字节(Byte)=8 位(bit)

1KB(Kilobyte，千字节)=1024B

1MB(Megabyte，兆字节)=1024KB =1024*1024 B

1GB(Gigabyte，吉字节，千兆)=1024MB

1TB(Trillionbyte，万亿字节，太字节)=1024GB

5. 冒泡排序算法　答案 D

伪代码如下：

输入:数组 L, n≥1。
输出:按非递减顺序排序的 L。
算法 BubbleSort:

```
1.  FLAG←n  //标记被交换的最后元素位置
2.    while FLAG > 1 do
3         k ← FLAG-1
4         FLAG ← 1
5         for j=1 to k do
6              if L(j) > L(j+1) then do
7.             L(j) <-> L(j+1)
8.             FLAG ← j
```

对 n 个数用以上冒泡排序算法进行排序，最少需要比较多少次?()。
A. n B. n-2 C. n^2 D. n-1

冒泡排序：是一种最基础的交换排序

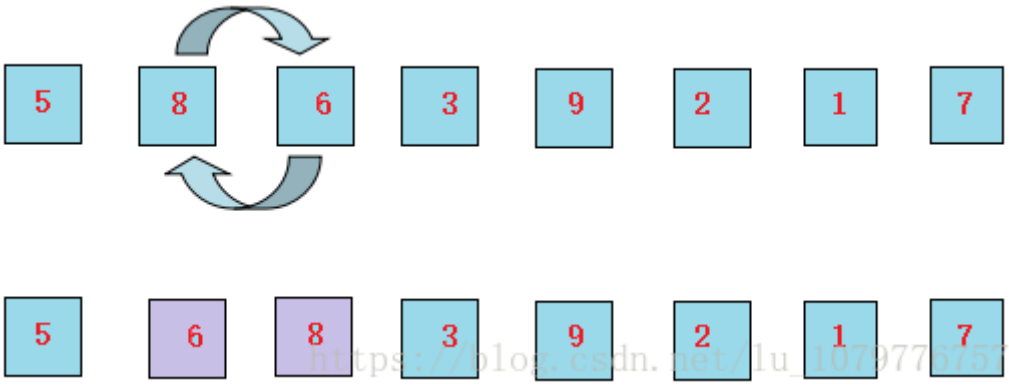
大家一定都喝过汽水吧，汽水中常常有许多小小的气泡，往上飘，这是因为组成小气泡的二氧化碳比水要轻，所以小气泡才会一点一点的向上浮。而冒泡排序之所以叫冒泡排序，正是因为这种排序算法的每一个元素都可以向小气泡一样，根据自身大小，一点一点向着数组的一侧移动。具体如何移动呢？我们来看一下例子：

有 8 个数组组成一个无序数列：5,8,6,3,9,2,1,7，希望从大到小排序。按照冒泡排序的思想，我们要把相邻的元素两两进行比较，根据大小交换元素的位置，过程如下：

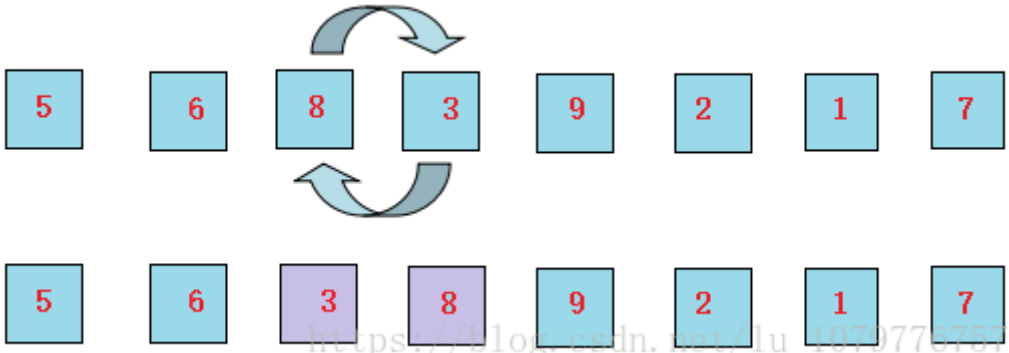
第一轮排序：

1、首先让 5 和 8 进行交换，发现 5 比 8 小，因此元素位置不变。

2、接下来让 8 和 6 比较，发现 8 比 6 大，所以要交换 8 和 6 的位置。3、

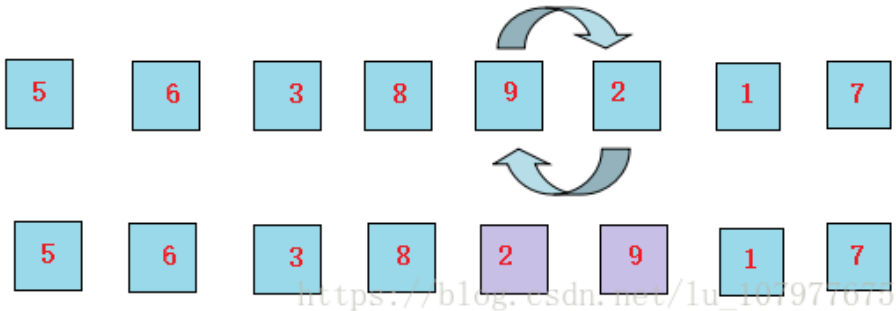


3、继续让 8 和 3 比较，发现 8 比 3 要大，所以 8 和 3 交换位置。

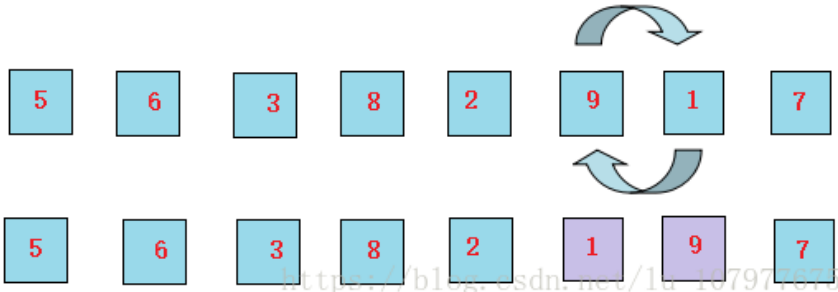


4、继续让 8 和 9 进行比较，发现 8 比 9 小，不用交换

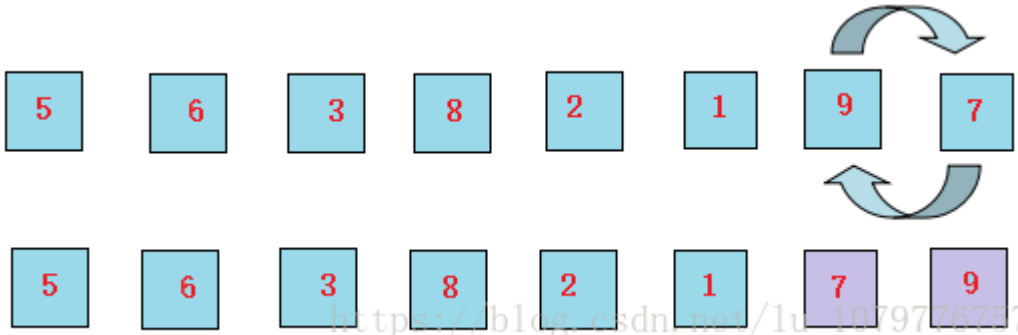
5、9 和 2 进行比较，发现 9 比 2 大，进行交换



6、继续让 9 和 1 进行比较，发现 9 比 1 大，所以交换 9 和 1 的位置。



7、最后，让 9 和 7 交换位置

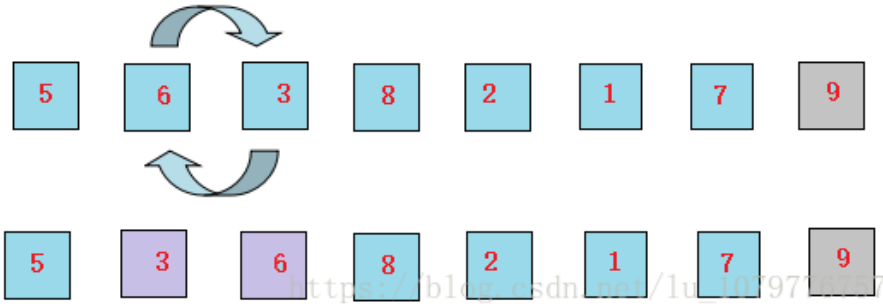


这样一来，元素 9 作为数列的最大元素，就已经排序好了。

下面我们来进行第二轮排序：

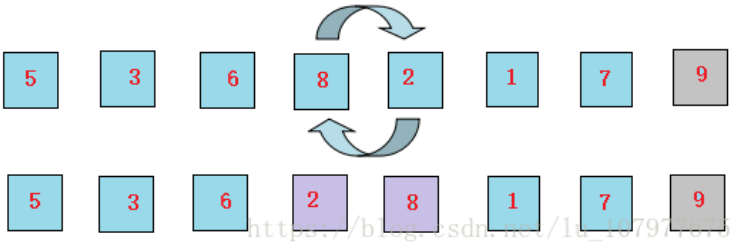
1、首先让 5 和 6 比较，发现 5 比 6 小，位置不变

2、接下来让 6 和 3 比较，发现 6 比 3 大，交换位置

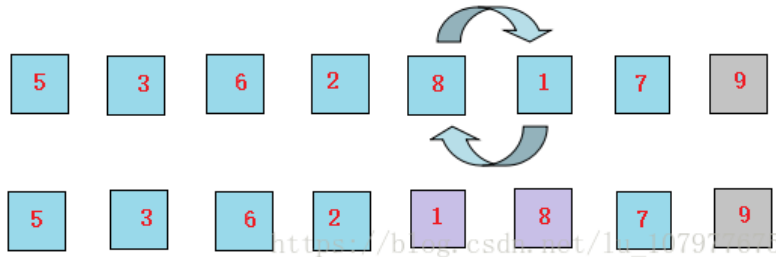


3、接下来让 6 和 8 比较，6 比 8 小，位置不变

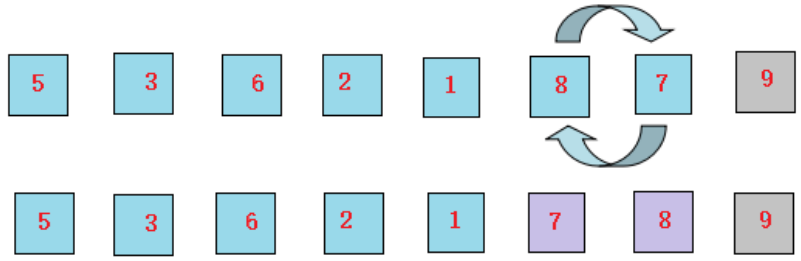
4、8 和 2 比较。8 比 2 大，交换位置



5、接下来让 8 和 1 比较，8 比 1 大，因此交换位置



6、继续让 8 和 7 比较，发现 8 比 7 大，交换位置



$$C_{\max} = \frac{n(n-1)}{2} = O(n^2)$$

$$M_{\max} = \frac{3n(n-1)}{2} = O(n^2)$$

6. 读程序（递归算法）答案 B

设 A 是 n 个实数的数组，考虑下面的递归算法:

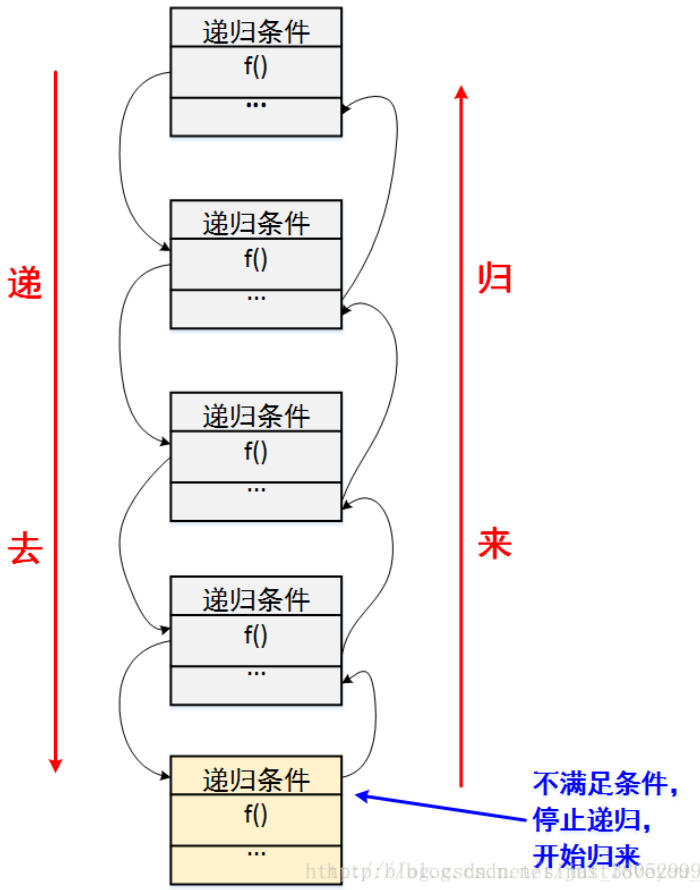
XYZ (A[1..n])

1. if n=1 then return A[1]
2. else temp ← XYZ(A[1..n-1])
3. if temp < A[n]
4. then return temp
5 else return A[n]

请问算法 XYZ 的输出是什么?(B)。

A.A 数组的平均 B.A 数组的最小值 C.A 数组的最大值. D.A 数组的中值

递归算法就是一个函数通过不断对自己的调用而求得最终结果的一种算法。



7. 链表 答案 B

链表不具有的特点是（ ）。

A.插入删除 不需要移动元素　B. 可随机访问任一元素

C.不必事 先估计存储空间　D. 所需空间与线性表长度成正比

链表的特点：动态申请内存（结点可以在运行时动态生成），内存率利用率高

插入、删除操作方便

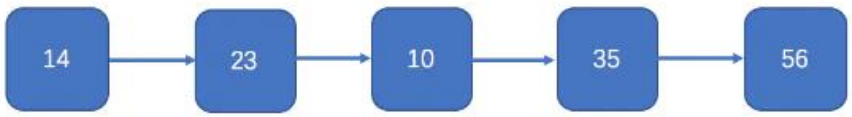
查询不方便。

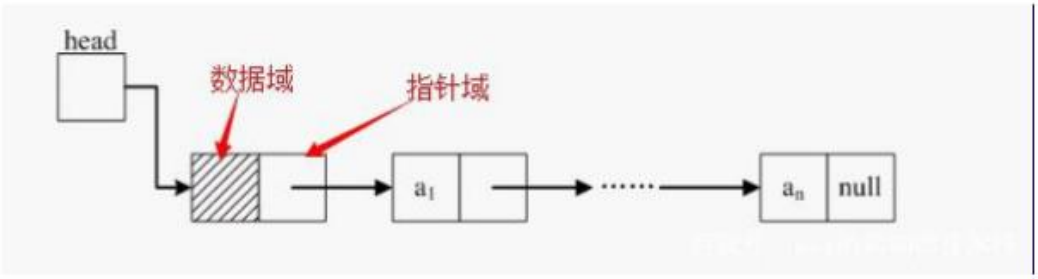
链表是一种物理存储单元上非连续、非顺序的存储结构，数据元素的逻辑顺序是通过链表中的指针链接次序实现的。**链表由一系列结点（链表中每一个元素称为结点）组成，结点可以在运行时动态生成。**每个结点包括两个部分：一个是存储数据元素的数据域，另一个是存储下一个结点地址的指针域。 相比于线性表顺序结构，操作复杂。

链表这类数据结构，有点像生活中的火车，一节车厢连着下一节车厢，在火车里面，只有到了 4 号车厢你才能进入 5 号车厢，一般情况下，不可能直接在 3 号车厢绕过 4 号车厢进入 5 号车厢。不过更准确来说，火车是双向链表，也就是说在 4 号车厢也可以反向进入 3 号车厢。

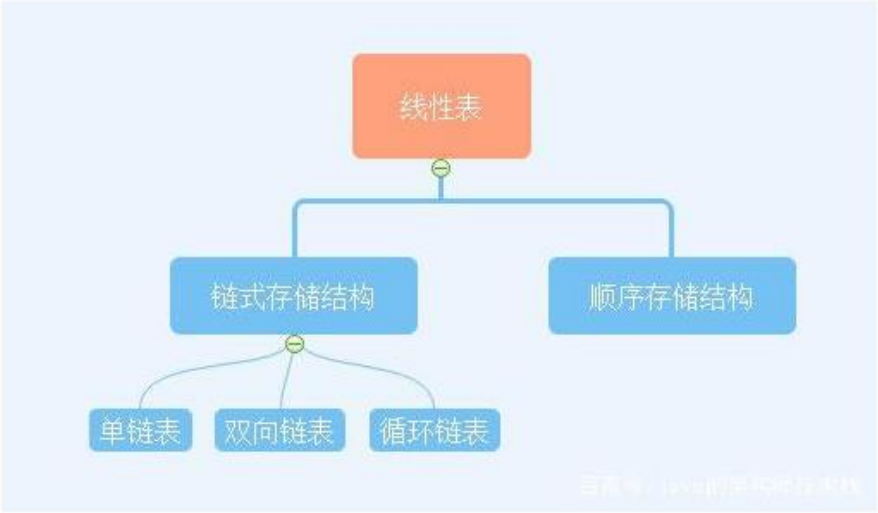
下面我们来画个图看看链表这类数据结构到底长啥样子。

直观感受一下链表数据结构



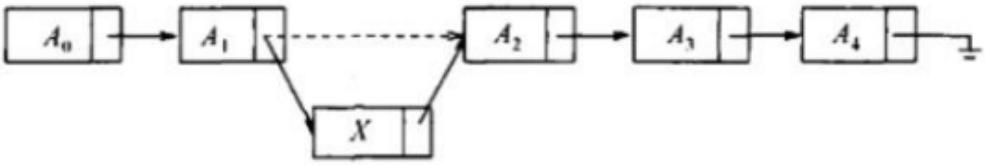


但是这只是基础，对线性表的描述，不想花费太多时间在这。下面用一张图表述吧。



第一个操作：插入（三种方式：头插入、尾插入、中间插入）

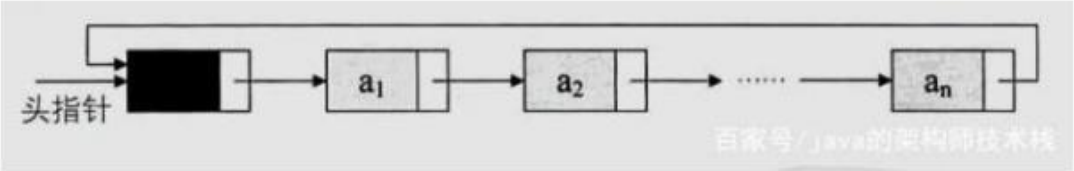
对于插入操作，使用一张图简单表述一下



插入：不需要移动元素　A：正确

第二种操作：删除

对删除操作同样一张图



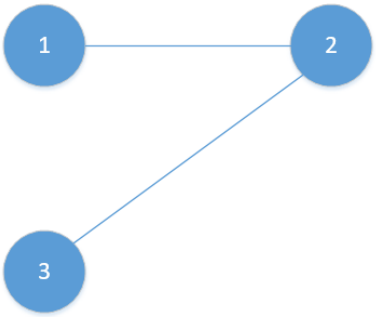
动态申请内存：不必事先估计存储空间　　B:正确
动态申请内存：存储空间与线性表的长度成正比　c：正确

访问：需要通过后向指针访问下一个元素，不可以随机访问任一元素。

8. 图 答案 C

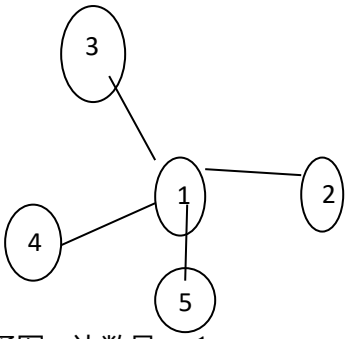
有 10 个顶点的无向图至少应该有(C)条边才能确保是一个连通图。
　　A.10　B.12 C.9　D.11

连通图：如果图中任意两点都是连通的，那么图被称作连通图。



连通图

星状连通图



n 个顶点星型连通图，边数是 n-1

9. 二进制 答案 C

二进制数 1011 转换成十进制数是（ ）。

A. 10 B.13 C.11 D.12

$(1011)_2$= 1*2^3 + 0*2^2 +1*2^1 + 1*2^0

=11

10. 排列组合 答案 D

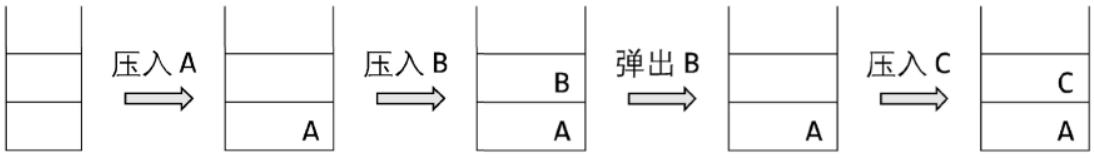
五个小朋友并排站成一列，其中有两个小朋友是双胞胎，如果要求这两个双胞胎必须相邻，则有(48) 种不同排列方法?

A.24 B.36 C.72 D. 48

注：两个双胞胎是一个单位，所以方案就是 4 的全排列 4!=24，然后双胞胎自已的全排列是 2,得数是 24*2=48

11. 栈 答案 C

下图中所使用的数据结构是(C) 。



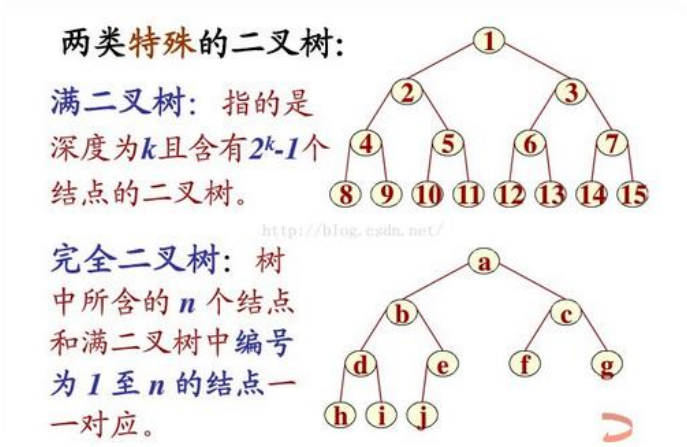
A. 哈希表 B.二叉树 C. 栈 D. 队列

注：栈 后进先出

12. 完全二叉数 答案 D

独根树的高度为 1。具有 61 个结点的完全二叉树的高度为(D)。

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



完全二叉树：设二叉树的深度为h，除第 h 层外，其它各层 (1～h-1) 的结点数都达到最大个数，第 h 层所有的结点都连续集中在最左边

满二叉树：深度为k且有2^k-1个结点的二叉树称为满二叉树

2^6-1=63 >61　2^5-1= 31<61　　树的高度是 6

13. 余数　答案 B

干支纪年法是中国传统的纪年方法，由 10 个天干和 12 个地支组合成 60 个天干地支。由公历年份可以根据以下公式和表格换算出对应的天干地支。

天干=（公历年份）除以 10 所的余数

地支=（公历年份）除以 12 所的余数

天干	甲	乙	丙	丁	戊	己	庚	辛	壬	癸		
	4	5	6	7	8	9	0	1	2	3		
地支	子	丑	寅	卯	辰	巳	午	未	申	酉	戌	亥
	4	5	6	7	8	9	10	11	0	1	2	3

例如，今年是 2020 年，2020 除以 10 余数为 0，查表为“庚”；2020 除以 12，余数为 4，查表为“子”，所以今年是庚子年。

请问 1949 年的天干地支是（ ）

A. 己亥　　B.己丑　C. 己卯　　D.己酉

14. 排列组合 答案 B

10 个三好学生名额分配到 7 个班级，每个班级至少有一个名额，一共有(B)种不同的分配方案。

A.56　B.84　C.72　D.504

注：一共 10 个名额，7 个班级，每个班级至少一个，去掉 7 个，

10-7=3 ，

7 个班级分 3 个名额：分为，1，2；　2,1；　1，1，1 ；　3

1，2 = 7*6/2=21　: C(6,2)

2，　1= 7*6/2=21　: C(6,2)

1,1,1 = 7*6*5 /(3*2*1) = 35 : C(7,3)

3=7　: C(7,1)

21+21+35+7 = 84

15. 排列组合 答案 D

有五副不同颜色的手套(共 10 只手套,每副手套左右手各 1 只),一次性从中取 6 只手套,请问恰好能配成两副手套的不同取法有(D) 种。

A.30　B.150　C.180　D.120

注：五副手套先选出 4 只两副：　C(5,2)

6 只中再选出 2 只：C（6，2）

6 只中选出一副的选法 C(3,1)

共有选法：C(5,2)*(C(6,2)-C(3,1)) = 120

二、阅读程序

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 v,错误填 x;除特殊说明外，判断题 1.5 分，选择题 3 分，共计 40 分)

1. 阅读程序 1

```cpp
#include <cstdlib>
#include <iostream>
using namespace std;

char encoder[26] = {'C', 'S', 'P', 0};
char decoder[26];

string st;

int main() {
    int k = 0;
    for (int i = 0; i < 26; ++i)
        if (encoder[i] != 0) ++k;
    for (char x = 'A'; x <= 'Z'; ++x) {
        bool flag = true;
        for (int i = 0; i < 26; ++i)
            if (encoder[i] == x) {
                flag = false;
                break;
            }
        if (flag) {
            encoder[k] = x;
            ++k;
        }
    }
    for (int i = 0; i < 26; ++i)
        decoder[encoder[i] - 'A'] = i + 'A';
    cin >> st;
    for (int i = 0; i < st.length(); ++i)
        st[i] = decoder[st[i] - 'A'];
    cout << st;
    return 0;
}
```

解析：

代码：12-13 :求 K=3

代码：12-25 行

数组 encoder[26]赋值： C S P A B D E F G H I J K L M N O Q R T U V W X Y Z

代码：26-26 行

数组 decorder[26]赋值：D E A F G H I J K L M N O P Q C R S B T U V W X Y Z

代码：29-30 行

St 值变更

判断题

1)输入的字符串应当只由大写字母组成，否则在访问数组时可能越界。(true)

小写字母与对应大写字母的 ASCII 相差 32，如果输入小写字母，decorder 长度是 26，32 就越界了。

2)若输入的字符串不是空串，则输入的字符串与输出的字符串一定不一样。(false)

输入索引位置与字母顺序相同的时输入与输出相同。

比如输入 XX 输出的是 XX

3)将第 12 行的“i < 26”改为“i < 16”，程序运行结果不会改变。(true)

因为 encorder 的长度是 3，所以不会改变运行结果。K 的值都是 3

4)将第 26 行的“i < 26”改为“i < 16”，程序运行结果不会改变。(false)

i 的范围影响 decorder 的赋值。

单选题

5)若输出的字符串为“ABCABCABCA"，则下列说法正确的是(C) 。

A.输入的字符串中既有 A 又有 P B.输入的字符串中既有 S 又有 B

C.输入的字符串中既有 S 又有 P D.输入的字符串中既有 A 又有 B

输入：CSPCSPCSPC 输出：ABCABCABCA

6)若输出的字符串为“CSPCSPCSPCSP”，则下列说法正确的是(D) 。

A. 输入的字符串中既有 J 又有 R B. 输入的字符串中既有 P 又有 K

C. 输入的字符串中既有 J 又有 K D. 输入的字符串中既有 P 又有 R

输入：PRN 输出：CSP

2. 阅读程序 2

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

long long n, ans;
int k, len;
long long d[1000000];

int main() {
    cin >> n >> k;
    d[0] = 0;
    len = 1;
    ans = 0;
    for (long long i = 0; i < n; ++i) {
        ++d[0];
        for (int j = 0; j + 1 < len; ++j) {
            if (d[j] == k) {
                d[j] = 0;
                d[j + 1] += 1;
                ++ans;
            }
        }
        if (d[len - 1] == k) {
            d[len - 1] = 0;
            d[len] = 1;
            ++len;
            ++ans;
        }
    }
    cout << ans << endl;
    return 0;
```

假设输入的 n 是不超过 2^62 的正整数，k 都是不超过 10000 的正整数，完成下面的判断题和单选题:

判断题

1)若 k=1，则输出 ans 时，len=n。(false)

反例:

n=1,k=1 : ans =0, len=2

n=3 ,k=1: ans =3 ,Len=2;

2)若 k>1,则输出 ans 时，len 一定小于 n。 (false)

反例：n=2 k=2;
ans=1, len=2; len =n

3)若 k>1，则输出 ans 时，k^len 一定大于 n。(true)

Len 的初始值是 1，
K^len > n

单选题
4)若输入的 n 等于 10^15,输入的 k 为 1，则输出等于(D) 。
A.(10^30-10^15)/2 B.(10^30+10^15)/2 C.1 D.10^15

K=1 时，ans = n

5)若输入的 n 等于 205,891,132,094,649(即 3^30)，输入的 k 为 3，则输出等于(A)。
A.(3^30-1)/2 B.3^30 C.3^30-1 D. (3^30+1)/2
总结：
n=3,ans=1; =(3^1-1)/2 =3/3
n=9 ,ans=4 =(3^2 -1)/2 =9/3 + 3/3
n=27, ans =13 =(3^3 – 1)/2 =27/3+9/3+3/3=13
n=3^m ,ans=(3^m-1) /2 =3^m/3 + 3^(m-1)/3 +….+3/3

6)若输入的 n 等于 100, 010, 002,000 , 090， 输入的 k 为 10， 则输出等于(D)
A.11,112,222,444,543 B.11,122, 222 ,444 ,453
C.11,122,222,444,543 D.11,112 ,222, 444 ,453

```
ans = 1 0 0, 0 1 0, 0 0 2, 0 0 0 , 0 9
        1 0 0, 0 1 0, 0 0 2, 0 0 0 , 0
          1 0 0, 0 1 0, 0 0 2, 0 0 0
            1 0 0, 0 1 0, 0 0 2, 0 0
              1 0 0, 0 1 0, 0 0 2, 0
                1 0 0, 0 1 0, 0 0 2
                  1 0 0, 0 1 0, 0 0
                    1 0 0, 0 1 0, 0
                      1 0 0, 0 1 0
                        1 0 0, 0 1
                          1 0 0, 0
                            1 0 0
                              1 0
                                1
```

3. 阅读程序 3

```
#include <algorithm>
#include <iostream>
using namespace std;

int n;
int d[50][2];
int ans;

void dfs(int n, int sum) {
    if (n == 1) {
        ans = max(sum, ans);
        return;
    }
    for (int i = 1; i < n; ++i) {
        int a = d[i - 1][0], b = d[i - 1][1];
        int x = d[i][0], y = d[i][1];
        d[i - 1][0] = a + x;
        d[i - 1][1] = b + y;
        for (int j = i; j < n - 1; ++j)
            d[j][0] = d[j + 1][0], d[j][1] = d[j + 1][1];
        int s = a + x + abs(b - y);
        dfs(n - 1, sum + s);
        for (int j = n - 1; j > i; --j)
            d[j][0] = d[j - 1][0], d[j][1] = d[j - 1][1];
        d[i - 1][0] = a, d[i - 1][1] = b;
        d[i][0] = x, d[i][1] = y;
    }
}

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

假设输入的 n 是不超过 50 的正整数，d[i][0]、 d[i][1]都是不超过
10000 的正整数，完成下面的判断题和单选题:

判断题
1)若输入 n 为 0，此程序可能会死循环或发生运行错误。(false)

n=0 ,主函数与子函数的代码都不执行，不会发生死循环或发生运行错误。

2)若输入 n 为 20，接下来的输入全为 0，则输出为 0。(true)

n=20 数组的值都是 0
ans 的值也是 0

3)输出的数一定不小于输入的 d[i][0]和 d[i][1]的任意一个。(false)

n=1, 1,1,
输出数：0

单选题
4)若输入的 n 为 20，接下来的输入是 20 个 9 和 20 个 0，则输出为(C)。
A.1917 B.1908 C. 1881 D.1890
n=1 ans=0
n=2 ans=2*9
n=3 ans=5*9=(2+3)*9
n=4 ans=9*9=(2+3+4)*9

n=m ans =(2+3+...+m)*9
=(1+2+..+m)*9 -9
=m*(m+1)/2*9 -9
n=20 ans=1881

5)若输入的 n 为 30，接下来的输入是 30 个 0 和 30 个 5，则输出为(B)。
A.2020 B.2030 C. 2010 D.2000

n=1 ans=0

n=2 ans=0

n=3 ans=1*5=(3-2)*5

n=4 ans=3*5=（1+n-2）*5

n=5 ans=6*5=(1+2+n-2)*5

n=6 ans=10*5 =(1+2+3+n-2) = (n-1)*(n-2)/2*5 = 2030

n=m ans = (1+2+3+...+m-2)*5=(m-1)*(m-2)/2*5

6) (4 分)若输入的 n 为 15，接下来的输入是 15 到 1，以及 15 到 1，则输出为(D)。

A. 2420 B.2220 C. 2440 D.2240

方法 1：

先求左边：

n=1 ans=0

n=2 ans=3

n=3 ans=11=(3*2)+(1^2+2^2)=11

n=4 ans=26=(3*4)+(1^2+2^2+3^2)=16

右边：

n=1 ans=0

n=2 ans=1

n=3 ans=5=(1^2 + 2^2)

n=4 ans=14(1^2+2^2+...+(n-1)^2)

左边+右边= n*(n-1)+2(1^2+2^2+...+(n-1)^2)

方法 2：通过小数找规律

n=1 ans=0;

n=2 ans=4 = 2(1 + 1*1)

n=3 ans=16 = 2(1+2+1*1 + 2*2)

n=4 ans =40=2（1+2+3 +1*1+2*2+3*3)

n=5 ans=80=2（1+2+3+4+1*1+2*2+3*3+4*4)

n=6 ans=140=2（1+2+3+4+5+1*1+....+5*5)

n=7　ans =224 =2(1+2+3+4+..+6+1*1+....6*6)

n : ans =2(1+...+n-1 + 1*1+2*2+...+(n-1)*(n-1))

三、完善程序

1. 完善程序 1

(质因数分解)给出正整数 n,请输出将 n 质因数分解的结果，结果从小到大输出。
例如:输入 n=120,程序应该输出 2 2 2 3 5，表示 120=2*2*2*3*5。
输入保证 2≤n≤10^9。
提示:先从小到大枚举变量 i，然后用 i 不停试除 n 来寻找所有的质因子。
试补全程序。

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

int n, i;

int main() {
    scanf("%d", &n);
    for (i = ①; ② <= n; i ++) {
        ③ {
            printf("%d ", i);
            n = n / i;
        }
    }
    if (④) {
        printf("%d ", ⑤);
    }
    return 0;
}
```

1)处应填(D)

A.n-1 B. 0 C.1 D.2

求因子，从 2 开始找。循环的初值是 2

2)处应填(D)

A.n/i B. n /(i*i) C. i*i*i D. i*i

求因子，以开平方根为边界，因子正常是左边一个，右边一个。

3)处应填(D)

A.if (i*i<=n) B.if(n%i==0) C.while(i*i<=n) D.while(n%i==0)

如果整除余数为 0，就是倍数关系。

4)处应填(A)

A.n>1 B.n<= 1 C.i+i<=n D. i < n/i

以 n 平方根为界，因子左边一个，右边一个。

我们循环的条件是 i*i<n

可以举例：6=2*3

循环到 i=2,当 i=3 时跳出循环。6/2 =3

最后一步：n>1 时输出 n

5)处应填(D)

A. 2 B. i C.n/i D. n

结合第 4 题，来做第 5 题。

2. 完善程序 2

(最小区间覆盖)给出 n 个区间，第 i 个区间的左右端点是[a_i,b_i]。现在要在这些区间中选出若干个，使得区间[0, m]被所选区间的并覆盖(即每一个 0≤i≤m 都在某个所选的区间中)。保证答案存在，求所选区间个数的最小值。

输入第一行包含两个整数 n 和 m (1≤n≤5000,1≤m≤10^9)。

接下来 n 行，每行两个整数 ai, bi; (0≤ai, bi≤m)。

提示:使用贪心法解决这个问题。先用θ(n2)的时间复杂度排序，然后贪心选择这些区间。

试补全程序。

```
#include <iostream>

using namespace std;

const int MAXN = 5000;
int n, m;
struct segment { int a, b; } A[MAXN];

void sort() // 排序
{
    for (int i = 0; i < n; i++)
        for (int j = 1; j < n; j++)
            if (①)
            {
                segment t = A[j];
                ②
            }
}
```

```
int main()
{
    cin >> n >> m;
    for (int i = 0; i < n; i++)
        cin >> A[i].a >> A[i].b;
    sort();
    int p = 1;
    for (int i = 1; i < n; i++)
        if (③)
            A[p++] = A[i];
    n = p;
    int ans = 0, r = 0;
    int q = 0;
    while (r < m)
    {
        while (④)
            q++;
        ⑤;
        ans++;
    }
    cout << ans << endl;
    return 0;
}
```

1)处应填(C)

A. A[j].b < A[j - 1].b　B. A[j].b > A[j - 1].b

C. A[j].a < A[j - 1].a　D. A[j].a > A[j - 1].a

以元素 a 将数组进行升序排序，

2)处应填(C)

A.A[j-1]=A[j];A[j]=t;

B.A[j+1]=A[j];A[j]=t;

C.A[j]=A[j-1];A[j-1]=t;

D.A[j]=A[j+1];A[j+1]=t;

J-1 与 J 位置进行交换。

3)处应填(C)

A. A[i].b < A[p - 1].b　　B. A[i].b > A[i - 1].b

C. A[i].b > A[p - 1].b　　D. A[i].b < A[i - 1].b

查找边界值，根据元素的 b 的值小于 A[p-1].b 时停止。

4)处应填(B)
A.q+1 < n && A[q+1].b<=r
B.q+1 < n && A[q+1].a<=r
C.q < n && A[q].a<=r
D.q < n && A[q].b<=r

查找边界范围

5)处应填(B)
A. r = max(r, A[q + 1].a) B. r = max(r, A[q].b)
C. r = max(r, A[q + 1].b) D. q++

找出 r 的最大值。