/

第十一届全国青少年信息学奥林匹克联赛

初赛试题分析

一、 单项选择题 （共 10 题，每题 1.5 分，共计 15 分。每题有且仅有一个正确答案.）。

（BADED，EEBAC）

1. 字符串“ababacbab”和字符串“abcba”的最长公共子串是（ B ）。
A. abcba B. cba C. abc D. ab E. bcba

四个选项从长到短一个个验证下就行了

2. 设全集 I = {a, b, c, d, e, f, g, h}，集合 $A \cup B$ = {a, b, c, d, e, f}， $A \cap C$ = {c, d, e}， $A \cap \sim B$ = {a, d}，那么集合 $A \cap B \cap C$ 为（ ）。
A. {c, e} B. {d, e} C. {e} D. {c, d, e} E. {d, f}

画张韦恩图搞定

3. 以下二进制数的值与十进制数 23.456 的值最接近的是（ D ）。
A. 10111.0101 B. 11011.1111 C. 11011.0111 D. 10111.0111 E. 10111.1111

实际考的是十进制数和二进制数的转化，参看去年的试题分析
整数部分 23（10）＝10111（2）
小数部分转化的方法，不断的把小数部分乘以 2，把整数部分的值记录下来就是相应的二进制小数位上的数。
0.456*2=0.912，记录 0；.912*2=1.824，记录 1；0.824*2＝1.648，记录 1；0.648*2=1.296，记录 1。

4. 完全二叉树的结点个数为 4 * N + 3，则它的叶结点个数为（ E ）。
A. 2 * N B. 2 * N - 1 C. 2 * N + 1 D. 2 * N - 2 E. 2 * N + 2

最简单的做法，代入特殊值。满二叉树是特殊的二叉树，满二叉树的所有结点数是叶子结点数的 2 倍减 1。
死算：设完全二叉树除去最后一层，有x层，最后一层有y个结点。那么前x层有 2^{x+1}－1 个结点，加上最后一层y个结点，得到：2^{x+1}-1+y=4N+3，所以 2^{x+1}+y=4N+4。由于最后一层有y个结点，所有最后第二层的前y/2 个结点有孩子，后 2^{x}-y/2 个结点是叶子结点，所以总的叶子结点是 2^{x}-y/2+y=2^{x}+y/2=2N+2。

/

5. 平面上有五个点 A(5, 3), B(3, 5), C(2, 1), D(3, 3), E(5, 1)。以这五点作为完全图 G 的顶点，每两点之间的直线距离是图 G 中对应边的权值。图 G 的最小生成树中的所有边的权值综合为（ D ）。

A. 8 B. $7+\sqrt{5}$ C. 9 D. $6+\sqrt{5}$ E. $4+2\sqrt{2}+\sqrt{5}$

首先列张表把两两之间的距离列出来

	A	B	C	D	E
A	#	$2\sqrt{2}$	$\sqrt{13}$	2	2
B	$2\sqrt{2}$	#	$\sqrt{17}$	2	$2\sqrt{5}$
C	$\sqrt{13}$	$\sqrt{17}$	#	$\sqrt{5}$	3
D	2	2	$\sqrt{5}$	#	$2\sqrt{2}$
E	2	$2\sqrt{5}$	3	$2\sqrt{2}$	#

最小生成树的算法，把边排下序，每次选择最小的，而且不会构成环的边添加到树中，添加 n-1 次结束（Kruskal 算法）。

排序后得到 2，2，2，$\sqrt{5}$，$2\sqrt{2}$，$2\sqrt{2}$，$2\sqrt{5}$，3，$\sqrt{13}$，$\sqrt{17}$

结果发现前四条边正好是一个最小生成树。

6. 下列设备中没有计算功能的是（ E ）。
A. 笔记本电脑 B. 掌上电脑 C. 智能手机
D. 电子计算器 E. 液晶显示器

7. Intel 的首颗 64 位处理器是（ E ）。
A. 8088 B. 8086 C. 80386 D. 80486 E. Pentium

8. 常见的邮件传输服务器使用（ B ）协议发送邮件。
A. HTTP B. SMTP C. TCP D. FTP E. POP3

9. 不能在 Linux 上使用的网页浏览器是（ A ）。
A. Internet Explore B. Netscape C. Opera D. Firefox E. Mozilla

10. 一位艺术史学家有 20000 幅 1024 * 768 的真彩色图像，如果将这些图像以位图形式保存在 CD 光盘上（一张 CD 光盘的容量按 600M 计算），大约需要（ C ）张 CD 光盘。
A. 1 B. 10 C. 100 D. 1000 E. 10000

分析：

这里没说是 24 位真彩色还是 32 位真彩色，不过这里只要求大约，答案

/

也是数量级的区别，所以随便拿 24 或者 32 算好了。比如 24 位，每张图片的大小是：1024*768*3 大约是 2 的 21 次方，是 2M。总共是 40000M，除以 600M/张，大概是 100。

二、 不定项选择题 （共 10 题，每题 1.5 分，共计 15 分。多选或少选均不得分）。

（CDE BCE BC CE BCE，CDE ACD BCDE ABCDE BDE）

11. 设 A = true，B = false，C = false，D = true，以下逻辑运算表达式值为真的有（ CDE ）。
A. (A∧B)∨(C∧D) B. ((A∧B)∨C)∧D C. A∧((B∨C)∨D)
D. (A∧(B∨C))∨D E. (A∨B)∧(C∨D)

12. $(3725)_8$ + $(B)_{16}$的运算结果是（ BCE ）。
A. $(3736)_8$ B. $(2016)_{10}$ C. $(11111100000)_2$ D. $(3006)_{10}$ E. $(7E0)_{16}$

13. 二叉树 T 的宽度优先遍历序列为 A B C D E F G H I，已知 A 是 C 的父结点，D 是 G 的父结点，F 是 I 的父结点，树中所有结点的最大深度为 3（根结点深度设为 0），可知 E 的父结点可能是（ BC ）。
A. A B. B C. C D. D E. F

A 直接排除，A 只有 BC 两个孩子
D 也排除 如果 D 有孩子 E，而且有孩子 G，那么 F 也是 D 的孩子，和二叉树矛盾
E 也排除 父亲结点在宽度优先遍历中一定比孩子先出现。
BC 试下就知道可以了。

14. 设栈 S 的初始状态为空，元素 a, b, c, d, e, f, g 依次入栈，以下出栈序列不可能出现的有（ CE ）。
A. a, b, c, e, d, f, g B. b, c, a, f, e, g, d C. a, e, c, b, d, f, g
D. d, c, f, e, b, a, g E. g, e, f, d, c, b, a

一个个验证就行了
参看第 9 届的 19 题分析，知道 C 问题出在 ebd，E 问题出在 gef

15. 下列外设接口中可以通过无线连接的方式连接设备的是（ BCE ）。
A. USB 2.0 高速版 B. 红外 C. 蓝牙 D. 串口 E. IEEE 802.11g 无线网卡

16. 处理器A每秒处理的指令数是处理器B的 2 倍。某一特定程序P分别编译为处理器A和处理器B的指令，编译结果处理器A的指令数是处理器B的 4 倍。

/

已知程序P的算法时间复杂度为$O(n^2)$，如果处理器A执行程序P时能在一小时内完成的输入规模为n，则处理器B执行程序P时能在一小时内完成的输入规模为（ CDE ）。

A. 4 * n　B. 2 * n　C. n　D. n / 2　E. n / 4

根据“处理器 A 每秒处理的指令数是处理器 B 的 2 倍。某一特定程序 P 分别编译为处理器 A 和处理器 B 的指令，编译结果处理器 A 的指令数是处理器 B 的 4 倍。”知道，A 速度是 B 的一半，那么 A 在一小时内完成规模是 n，B 在一小时内完成的规模是$\sqrt{2}$ n。

17. 以下哪个（些）不是计算机的输出设备（ ACD ）。

A. 鼠标　B. 显示器　C. 键盘　D. 扫描仪　E. 绘图仪

18. 以下断电之后将不能保存数据的有（ BCDE ）。

A. 硬盘　B. 寄存器　C. 显存　D. 内存　E. 高速缓存

19. 下列活动中属于信息学奥赛系列活动的是（ ABCDE ）。

A. NOIP　B. NOI　C. IOI　D. 冬令营　E. 国家队选拔赛

、

20. 下列关于高级语言的说法正确的有（ BDE ）。

A. Ada 是历史上的第一个高级语言

B. Pascal 和 C 都是编译执行的高级语言

C. C++是历史上的第一个支持面向对象的语言

D. 编译器将高级语言程序转变为目标代码

E. 高级语言程序比汇编语言程序更容易从一种计算机移植到另一种计算机上

第一个高级语言是 fortan

第一个面向对象的语言是 smalltalk

三．问题求解（请在空格处填上答案，每空 5 分，共计 10 分）

1. 将数组{32, 74, 25, 53, 28, 43, 86, 47}中的元素按从小到大的顺序排列，每次可以交换任意两个元素，最少需要交换_____次。

答案：5

原来序列 32, 74, 25, 53, 28, 43, 86, 47

目标序列 25, 28, 32, 43, 47, 53, 74, 86

只要从小到大把那些数依次放到目标位置就行了。

2. 取火柴游戏的规则如下：一堆火柴有 N 根，A、B 两人轮流取出。每人每

/

次可以取 1 根或 2 根，最先没有火柴可取的人为败方，另一方为胜方。如果先取者有必胜策略则记为 1，先取者没有必胜策略记为 0。当 N 分别为 100，200，300，400，500 时，先取者有无必胜策略的标记顺序为（回答应为一个由 0 和/或 1 组成的字符串）。

答案：11011

除以 3 的余数不是 0 就有必胜策略。

四．阅读程序（共 4 题，每题 8 分，共计 32 分）

1.

```
var
   a, b, c, p, q : integer;
   r : array[0..2] of integer;
begin
   read(a, b, c);
   p := a div b div c;
   q := b - c + a + p;
   r[0] := a * p div q * q;
   r[1] := r[0] * (r[0] - 300);
   if (3 * q - p mod 3 <= r[0]) and (r[2] = r[2]) then
      r[1] := r[r[0] div p mod 2]
   else r[1] := q mod p;
   writeln(r[0] - r[1]);
end.
```

输入：100 7 3

输出：_____

答案：－7453

现在必考的四则混合运算题，仔细做，做两遍，中间至少间隔半小时，错了别回来。

2.

```
var
   a : array [1..50] of integer;
   n, i, sum : integer;
procedure work(p, r: integer);
var
   i, j, temp : integer;
```

```
/
begin
   if p < r then begin
      i := p - 1;
      for j := p to r - 1 do
         if a[j] >= a[r] then begin
            inc(i);
            temp := a[i]; a[i] := a[j]; a[j] := temp;
         end;
      temp := a[i + 1]; a[i + 1] := a[r]; a[r] := temp;
      work(p, i);
      work(i + 2, r);
   end;
end;
begin
   read(n);
   for i := 1 to n do read(a[i]);
   work(1, n);
   for i := 1 to n - 1 do sum := sum + abs(a[i + 1] - a[i]);
   writeln(sum);
end.
```

输入：10 23 435 12 345 3123 43 456 12 32 -100

输出：_____

答案：3223

首先要看出来它是快速排序，排序后的结果第一个是 3123，最后一个是-100，它求的式子展开中间都能约去，最后是 3123-(-100)=3223。

快速排序的基本思想是：n 个元素被分成三段(组):左段 left，右段 right 和中段 middle。中段仅包含一个元素。左段中各元素都小于等于中段元素，右段中各元素都大于等于中段元素。因此 left 和 right 中的元素可以独立排序，并且不必对 left 和 right 的排序结果进行合并。

3.

```
var
   str : string;
   len, i, j : integer;
   nchr : array [0..25] of integer;
   mmin : char;
begin
   mmin := 'z';
```

```
/
    readln(str);
    len := length(str);
    i := len;
    while i >= 2 do begin
       if str[i - 1] < str[i] then break;
       dec(i);
    end;
    if i = 1 then begin
       writeln('No result!');
       exit;
    end;
    for j := 1 to i - 2 do write(str[j]);
    fillchar(nchr, sizeof(nchr), 0);
    for j := i to len do begin
       if (str[j] > str[i - 1]) and (str[j] < mmin) then
          mmin := str[j];
       inc(nchr[ord(str[j]) - ord('a')]);
    end;
    dec(nchr[ord(mmin) - ord('a')]);
    inc(nchr[ord(str[i - 1]) - ord('a')]);
    write(mmin);
    for i := 0 to 25 do
       for j := 1 to nchr[i] do
          write(chr(i + ord('a')));
    writeln;
end.
```

输入：zzyzcccbbbaaa

输出：_____

答案：zzzaaabbbcccy

分析：
第一个循环是从后向前找第一个下降的地方，是 i=4 的时候。首先输出前两个字母“zz”。第三个循环中的 if 是第四位置开始找一个，比 y 大的最小的元素（而且比 mmin=z 小），所以这个 if 语句里面的执行语句是永远不会执行的，接下来是看 j 位置的数比 a 大几，就在 nchr 第几个位置增加 1，所以这个循环结束时 nchr[0]=3，nchr[1]=3，nchr[2]=3，nchr[25]=1。Dec 语句把 nchr[25]又变成 0 了，后面的 inc 语句把 nchr[24]变成 1。接下来 write(mmin)输出“z”，最后的循环看每个位置 nchr 的值，是几，把这个位置对应的字母输出几次，就是“aaabbbcccy”。

/

```
4. var
   n : longint;
function g(k : longint) : longint;
begin
   if k <= 1 then g := k
   else g := (2002 * g(k - 1) + 2003 * g(k - 2)) mod 2005;
end;
begin
   read(n);
   writeln(g(n));
end.
```

输入：2005

输出：_____

答案：31

分析：
首先看到，如果 g=（2002 * g(k - 1) + 2003 * g(k - 2)）得到的 g(2005)除以 2005 的余数就是本题的答案。还可以化简为求 g=（-3) * g(k - 1) + (-2)* g(k - 2))得到的 g(2005)除以 2005 的余数。下面是如何来求 g(2005)，已知递推式 g(n)=-3g(n-1)-2g(n-2)，求 g(n)的通项公式，可以利用特征方程来求：

一般地：如果数列 $\{a_n\}$ 具有如下的递推关系：
$a_n=c_1a_{n-1}+c_2a_{n-2}$(n=2，3，…)
则二次方程：$x^2=c_1x+c_2$为数列 $\{a_n\}$ 的特征方程。
设x_1，x_2是特征方程的两个根，我们有如下重要结论：

(1) 当$x_1\neq x_2$时，$a_n=\alpha_1x_1^n+\alpha_2x_2^n$（$\alpha_1$，$\alpha_2$是由初始条件确定的常数)。

(2) 当$x_1=x_2$时，$a_n=(\beta_1+\beta_2n)x_1^n$（$\beta_1$，$\beta_2$是由初始条件确定的常数)。

利用这个方法，我们来求 g(n)：
特征方程是 $x^2+3x+2=0$
两个根是 -1 和-2
由g(0)=0，g(1)=1，来确定α_1和α_2。
$\alpha_1(-1)^0+\alpha_2(-2)^0=0$
$\alpha_1(-1)^1+\alpha_2(-2)^1=1$
解得$\alpha_1=1$ $\alpha_2=-1$

/

所以g(n)的通项公式为$g(n)=(-1)^n+(-1)(-2)^n$

$g(2005)=2^{2005}-1$

下面的问题来求 $2^{2005}-1$ 除以 2005 的余数了，主要求 2^{2005} mod 2005

这里要用到数论中的几个结论：

（1）若 a1 mod b=c1，a2 mod b=c2，则(a1*a2) mod b=(c1*c2) mod b

（2）若 a mod b=m，a mod c=m，(b,c)=1，则 a mod (b*c)=m

（3）若a是一个整数，p是一个质数的话，那么a^p mod p=a（费马小定理）

若a不是p的倍数也可以写成a^{p-1} mod p=1

2^{2005} mod 401=32^{401} mod 401，由于 401 是质数，根据费马小定理知道结果是 32。

2^{2005} mod 5=$2^{6*334+1}$ mod 5=((64^{334} mod 5)*(2 mod 5)) mod 5=2 mod 5=32 mod 5

所以 2^{2005} mod 2005=32，所以结果是 32－1＝31。

对数学要求相当高的一道题，几乎没人在考场上做出来。

五．完善程序（前 5 空，每空 2 分，后 6 空，每空 3 分，共 28 分）

1．木材加工

题目描述:

木材厂有一些原木，现在想把这些木头切割成一些长度相同的小段木头（木头有可能有剩余），需要得到的小段的数目是给定的。当然，我们希望得到的小段越长越好，你的任务是计算能够得到的小段木头的最大长度。

木头长度的单位是 cm。原木的长度都是正整数，我们要求切割得到的小段木头的长度也是正整数。

输入:

第一行是两个正整数N和K(1 ≤ N ≤ 10000，1 ≤ K ≤ 10000)，N是原木的数目，K是需要得到的小段的数目。

接下来的N行，每行有一个1到10000之间的正整数，表示一根原木的长度。

输出:

输出能够切割得到的小段的最大长度。如果连1cm长的小段都切不出来，输出"0"。

输入样例:

```
3 7
232
124
456
```

输出样例:

/

114

程序：

```
var
   n, k : integer;
   len : array [1..10000] of integer;
   i, left, right, mid : integer;
function isok(t : integer) : boolean;
var
   num, i : integer;
begin
   num := 0;
   for i := 1 to n do begin
      if num >= k then break;
      num := ①;
   end;
   if ② then isok := true
   else isok := false;
end;
begin
   readln(n, k);
   right := 0;
   for i := 1 to n do begin
      readln(len[i]);
      if right < len[i] then right := len[i];
   end;
   inc(right);
   ③;
   while ④ < right do begin
      mid := (left + right) div 2;
      if ⑤ then right := mid
      else left := mid;
   end;
   writeln(left);
end.
```

答案：

（1）　num + len[i] div t

（2）　num >= k

/

（3）　Left := 0
（4）　Left + 1
（5）　not isok(mid) （或者 isok(mid) = false）

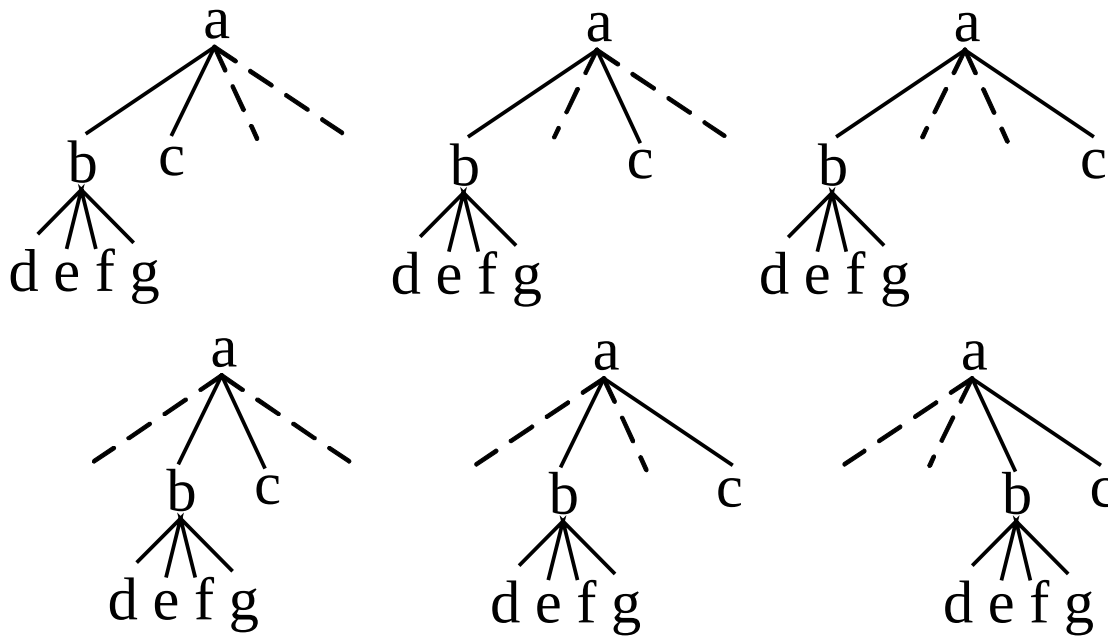
分析：
算法比较简单的一道题。如果自己想算法，肯定大家也想到枚举小木棍的长度，从大到小试就行了，如果正好段数是 k，那么就是我要的答案。那么能不能更快的知道小木棍的长度呢，看到代码里有 left，right，mid 的字样，懂二分查找的马上就应该明白它用二分查找在找小木棍的长度。Left 是 0，right 是最长的原木长度+1。明显 left 的值是肯定行，right 的值是肯定不行的。每次算中间 mid 的值，如果 mid 的值行，那么在 mid 和 right 中继续找；否则在 left 和 mid 中继续找，直到 left 超过 right。
下面可以读程序填空了，从主程序开始。第三个空是设置 left 为 0，实际这句不写也对，现在的编程软件不设初值都是 0，严格的说，这个空很多随便填的都对，你写 writeln(“Hello, teacher”)也不影响最后求出来的小木棍的长度。第四个空二分查找没结束的条件：一般来讲是 left<=right，但是一般的二分查找是 left=mid+1 或者 right=mid-1（参看以前的某道二分查找的选择题），现在代码中是 left=mid 和 right=mid，这样的话如果 left=right，那么 left 永远不会超过 right，由于本问题 left 是一定行的，right 是一定不行的，所以至少有一种行的情况（可能是 0），所以只要 left 和 right 靠一起，就肯定是 left 了，所以只有 left 和 right 中隔了数字才继续查找，所以是 left+1<right。第五个空就是什么时候是在前半段继续找？明显是 mid 的值不行，这就要去看 isok 函数了。第一个空明显是把一根根原木都去除以试的小木棍的长度，num 记录目前得到的所有小木棍的数量，num:=num+ + len[i] div t 。第二个空是啥时候行的，应该是 num>=k。所以第五个空是 not isok(mid)。

2．N 叉树

题目描述：
　　我们都了解二叉树的先根遍历，中根遍历和后根遍历。当知道先根遍历的结果和中根遍历结果的时候，我们可以唯一的确定二叉树；同样的，如果知道了后根遍历的结果和中根遍历结果，二叉树也是唯一确定的。但是如果只知道先根遍历和后根遍历的结果，二叉树就不是唯一的了。但是我们可以计算满足条件的不同二叉树一共有多少个。这不是一个很困难的问题，稍微复杂一点，我们把这个问题推广到 N 叉树。

　　我们用小写英文字母来表示 N 叉树的结点，不同的结点用不同的字母表示。比如，对于 4 叉树，如果先根遍历的结果是 abdefgc，后根遍历的结果是 defgbca，那么我们可以得到 6 个不同的 4 叉树（如下图）。

/



输入:

输入数据包括3行。

第一行是一个正整数N(2 ≤ N ≤ 20)，表示我们要考虑N叉树。

第二行和第三行分别是两个字符串序列,分别表示先根遍历和后根遍历的结果。

输出:

输出不同的N叉树的数目。题目中给的数据保证得到的结果小于2^{31}。

输入样例:

```
4
abdefgc
defgbca
```

输出样例:

```
6
```

程序:

```
var
   str1, str2 : string;
   N, len : integer;
   com : array[0..100, 0..100] of longint;
function getcom(x, y : integer) : longint;
begin
   if (y = 0) or (x = y) then ①
   else if com[x][y] <> 0 then getcom := com[x][y]
   else begin
```

```
/
      com[x][y] := getcom(x - 1, y)+ ②;
      getcom := com[x][y];
   end;
end;
function count(a, b, c : integer) : longint;
var
   sum : longint;
   k, s, t, p : integer;
begin
   sum := 1; k := 0; s := a + 1; t := c;
   if a = b then count := 1
   else begin
      while s <= b do begin
         p := t;
         while str1[s] <> str2[t] do inc(t);
         sum := sum * count(s, s + t - p, p);
         s := ③;
         ④; inc(k);
      end;
      count := ⑤ * getcom(N, k);
   end;
end;
begin
   readln(N); readln(str1); readln(str2);
   len := length(str1);
   writeln(count(⑥));
end.
```

答案：

（1） getcom := 1
（2） getcom(x - 1, y - 1)
（3） s + t - p + 1
（4） inc(t) （或者 t := t + 1）
（5） sum
（6） 1, len,1

分析：

自己先考虑一下算法，先序遍历的第一个结点是根结点，后序遍历的最后一

/

个结点是根结点，这样树的根知道了；那么先序的第二个结点就是第一个子树的根，那么第一个子树有哪些结点呢？明显在后序遍历中第一子树根结点前的结点都是，然后确定第二个子树的根结点和有哪些结点，依次类推，这样 n 叉树如果有 k（k<=n）个子树，我们还不能确定这 k 个子树在哪几个位置，既然求总数，就是所有的情况，那么就有 c(n,k)种，所以如果知道子树 1 的情况有 s1，子树 2 的情况有 s2，子树 k 的情况有 sk，那么该树总共有 s1*s2*s2……*sk*c(n,k)，所以算法就是递归加组合，递归的边界是只有一个结点的树是 1 种情况。

代码分析：getcom 是求组合数，x 就是我上面写的 n，y 就是 k，第一个空是递推的边界条件，第二个空是我以前讲过的组合数最重要的一个公式，其实就是递推公式。主要来分析它的 count 代码：根据前面自己设计的算法，看到代码里 if a=b then count=1，知道这是递归的边界条件，就是只有一个结点的树只有一种情况，那么 a,b 基本就是你目前求的这棵树的开始位置和结束位置，两者相等才是只有一个结点啊。由于自己设计算法中知道第一子树的根结点，要在后序遍历中找到这个结点，两者之间的结点是第一子树下的结点，看看代码中 str1[s]<str2[t]的比较，再看看前面 s=a+1，知道 s 基本是在先序遍历中查找的指示指针，t 基本是在后序遍历中查找的指示指针。那么程序中 while str1[s]<> str2[t] do inc(t)这句结束，第一子树的结点都确定了。看到 sum := sum * count(s, s + t - p, p);t-p 就是这次查找跑了几个元素，那么 s+t-p 就是第一子树的所有结点的最后一个结点的位置，p 说明是第一子树在后序中开始的位置，多长，可以根据它在先序遍历中从 s 开始到 s+t-p 结束算出来。那么在 while 循环中，每次把当前子树的情况数乘上去后，再去处理下一个子树，下一个开始的位置在 s+t-p+1，就是第三个空，同样在后序遍历中查找，要从刚才找到的那个根结点的下一个位置开始，t:=t+1，就是第四个空。第五个空就是根据乘法原理得到情况数后还要乘以组合数。最后第六个空就是在先序 1,len 中，在后序 1 开始处，递归求解。

代码中变量的说明：

count(a,b,c)表示在先序 a,b 范围内，后序 c 开始的长度是（b-a）的范围内能建立的树的总情况数。

s,t 是两个指针，分别在先序和后序中扫描，来确定每一个子树有哪些结点。

k 表示该树有几个子树。

p 用来保存 t 的出初始位置，这样 t 往后面扫到某个位置，根据 t-p 知道扫了几个元素，来确定该子树有多少结点。

getcom(x,y)求组合数 c(x,y)