2000 年全国青少年信息学（计算机）奥林匹克分区联赛复赛试题
（高中组 竞赛用时：3 小时）

题一 进制转换 (18 分)

问题描述:
我们可以用这样的方式来表示一个十进制数：将每个阿拉伯数字乘以一个以该数字所处位置的（值减 1）为指数，以 10 为底数的幂之和的形式。例如，123 可表示为 1*10^2+2*10^1+3*10^0 这样的形式。与之相似的，对二进制数来说，也可表示成每个二进制数码乘以一个以该数字所处位置的（值-1）为指数，以 2 为底数的幂之和的形式。一般说来，任何一个正整数 R 或一个负整数-R 都可以被选来作为一个数制系统的基数。如果是以 R 或-R 为基数，则需要用到的数码为 0，1,....R-1。例如，当 R=7 时，所需用到的数码是 0，1，2,
3，4，5 和 6，这与其是 R 或-R 无关。如果作为基数的数绝对值超过 10，则为了表示这些数码，通常使用英文字母来表示那些大于 9 的数码。例如对 16 进制数来说，用 A 表示 10，用 B 表示 11，用 C 表示 12，用 D 表示 13，用 E 表示 14，用 F 表示 15。在负进制数中是用-R 作为基数，例如-15（＋进制）相当于 110001（-2 进制），并且它可以被表示为 2 的幂级数的和数：
110001=1*(-2)^5+1*(-2)^4+0*(-2)^3+0*(-2)^2+0*(-2)^1+1*(-2)^0
问题求解:
设计一个程序，读入一个十进制数的基数和一个负进制数的基数，并将此十进制数转换为此负进制下的数：-R∈{-2，-3，-4,....-20}

输入:
输入的每行有两个输入数据。
第一个是十进制数 N（-32768<=N<=32767）；第二个是负进制数的基数-R。
输出:
结果显示在屏幕上，相对于输入，应输出此负进制数及其基数，若此基数超过 10，
则参照 16 进制的方式处理。

样例:
输入
30000 -2
-20000 -2
28800 -16
-25000 -16

输出
30000=11011010101110000(base -2)
-20000=1111011000100000(base -2)
28800=19180(base -16)
-25000=7FB8(base -16)

题二 乘积最大 (22 分)

问题描述:

今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰 90 周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一个好朋友 XZ 也有幸得以参加。活动中，主持人给所有参加活动的选手出了这样一道题目:

设有一个长度 N 的数字串，要求选手使用 K 个乘号将它分成 K+1 个部分，找出一种分法，使得这 K+1 个部分的乘积能够为最大。

同时，为了帮助选手能够正确理解题意，主持人还举了如下的一个例子：

有一个数字串: 312，当 N=3，K=1 时会有以下两种分法：

1）3*12=36
2）31*2=62

这时，符合题目要求的结果是： 31*2=62

现在，请你帮助你的好朋友 XZ 设计一个程序，求得正确的答案。

输入：
程序的输入共有两行:
第一行共有 2 个自然数 N,K (6<=N<=40，1<=K<=6)
第二行是一个 K 度为 N 的数字串。

输出:
结果显示在屏幕上，相对于输入，应输出所求得的最大乘积（一个自然数）。

样例:

输入
4 2
1231

输出
62

题三 单词接龙 (27 分)

问题描述：
单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙" 中出现两次），在两个单词相连时，其重合部分合为一部分，例如 beast 和 astonish，如果接成一条龙则变为 beastonish，另外相邻的两部分不能存在包含关系，例如 at 和 atide 间不能相连。

输入:
输入的第一行为一个单独的整数 n(n<=20）表示单词数，以下 n 行每行有一个单词，输入的最后一行为一个单个字符，表来“龙”开头的字母。你可以假定以此字母开头的“龙"一定存在。

输出:
只需输出以此字母开头的最长的“龙”的长度

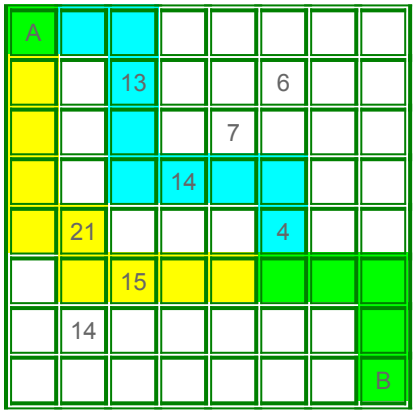
样例:

输入
5
at
touch
cheat
choose
tact
a

输出
23 （连成的“龙”为 atoucheatactactouchoose）

题四 方格取数 (33 分)

问题描述:
设有 N*N 的方格图(N<=8)，我们将其中的某些方格中填入正整数，而其他的方格中则放人数字 0。如下图所示（见样例 ，黄色和蓝色分别为两次走的路线，其中绿色的格子为黄色和蓝色共同走过的）：



某人从图的左上角的 A 点出发，可以向下行走，也可以向右走，直到到达右下角的 B 点。在走过的路上，他可以取走方格中的数（取走后的方格中将变为数字 0）。此人从 A 点到 B 点共走两次，试找出 2 条这样的路径，使得取得的数之和为最大。

输入:
输入的第一行为一个整数 N（表示 N*N 的方格图），接下来的每行有三个整数，前两个表示位置，第三个数为该位置上所放的数。一行单独的 0 表示输入结束。

输出:
只需输出一个整数，表示 2 条路径上取得的最大的和。

样例:
输入
8
2 3 13
2 6 6
3 5 7
4 4 14
5 2 21
5 6 4
6 3 15
7 2 14
0 0 0

输出
67

题一：进制转换（18 分）

问题描述：我们可以用这样的方法来表示一个十进制数：将每个阿拉伯数字乘以一个以该数字所处位置的（值减 1）为指数，以 10 为底的幂之和的形式。例如：123 可表示为 1*102+2*101+3*100 这样的形式。

与之相似的，对二进制来说，也可表示成每个二进制数码乘以一个以数字所处位置的（值-1）为指数，以 2 为底数的幂之和的形式。一般来说，任何一个正整数 R 或一个负整数-R 都可以被选来作为一个数制系统的基数。如果是以 R 或-R 为基数，则需要用到的数码为 0,1,.....R-1。例如当 R=7 时，所需用到的数码是 0,1,2,3,4,5 和 6，这与其是 R 或-R 无关。如果作为基数的数绝对值超过 10，则为了表示这些数码，通常使用英文字母来表示那些大于 9 的数码。例如对于 16 进制数来说，用 A 表示 10，用 B 表示 11，用 C 表示 12，和 D 表示 13，用 E 表示 14，用 F 表示 15。在负进制数中是用-R 作为基数，例如-15（十进制）相当于 110001（-2 进制），并且它可以被表示为 2 的幂级数的和数：

110001=1*(-2)5+1*(-2)4+0*(-2)3+0*(-2)2+0*(-2)1+1*(-2)0

问题求解：设计一个程序，读入一个十进制数和一个负进制数的基数，并将此十进制数转换为此负进制下的数：-R∈{-2,-3,-4,.....-20}

输入：输入的每行有两个输入数据。第一个是十进制数 N(-32767——32767)；第二个是负进制数的基数-R。

输出：结果显示在屏幕上，相对于输入，应输出此负进制及其基数，若此基数超过 10，则参照 16 进制的方式处理。

样例：输入：30000　　-2

　　-20000　　-2

　　28800　　-16

　　-25000　　-16

输出：30000=11011010101110000(base -2)

　　-20000=1111011000100000 (base -2)

　　28800=19180　(base -16)

　　-25000=7FB8　(base -16)

解：根据题意，-R 进制可表示为-R 的幂级数的和，转换时仍用除-R 取余的方法。转换后的数必须是正数，故用 N 除以-R 的余数不能是负数。使余数非负，是此题的关键。当余数为负时，就把商增大而使余数为正。下面以 22 转换成-3 进制为例作短除法。

故(22)10=(12021)-3

在程序处理时，当 N>=(N Div R)*R 时，余数为 N-(N Div R)*R，商为 N Div R；否则余数为 N-(N Div R+1)*R，商为 N Div R+1。TP 程序如下。

```
Const A:String[20] = '0123456789abcdefghij';

Var B:String[100];  N,R,Y,I,J,N1:Integer;

Begin Write('N,R='); Readln(N, R);  I:= 1; B[I] :='0'; N1:=N;

While N<>0 Do Begin

  If (N>=(N Div R)*R) Then    Begin Y:=N-(N Div R)*R; N:=N Div R End

  Else Begin Y:=N-(N Div R+1)*R; N:=N Div R+1; End;

  B[I]:=A[Y+1];  I:=I+1  End;

  Write(N1,'='); For J:= I Downto 1 Do Write(B[J]);  Writeln(' (Base ',R,')')

End.
```

题二：乘积最大（22 分）

问题描述：今年是国际数学联盟确定的“2000——世界数学年”，又恰逢我国著名数学家华罗庚先生诞辰 90 周年。在华罗庚先生的家乡江苏金坛，组织了一场别开生面的数学智力竞赛的活动，你的一位好朋友 XZ 也有幸得以参加。活动中，主特人给所有参加活动的选手出了这样一道 题目：

设有一个长度为 N 的字符串，要求选手使用 K 个乘号将它分成 K+1 个部分，找出一种分法，使得这 K+1 个部分的乘积能够为最大。

同时，为了帮助选手能够正确理解题意，主特人还举了如下的一个例子：

有一个数字串 312，当 N=3，K=1 时会有以下两种分法：

(1) 3*12=36　　(2) 31*2=62

这时，符合题目要求的是：31*2=62

现在，请你帮助你的好朋友 XZ 设计一个程序，求得正确的答案。

输入：程序的输入共有两行：第一行共有两个自然数 N，K(6<=N<=40，1<=K<=6)；第二行是一个长度为 N 的字符串。

输出：结果显示在屏幕上，相对于输入，应输出所求得的最大乘积(一个自然数)。

样例：输入：4　2

1231

输出：62

分析：很明显的动态规划。
令 d[i,j,k]为第 i 个数字到第 j 个数字加 k 个乘号所能够达到的最大值。
状态转移方程是：d[i,j,k]=max{num[i,t]*d[t+1,j,k-1]} (枚举 t, 满足 i<=t<=j-1)
注意到状态转移的时候总是使 k 减小（或不变），所以把 k 作为阶段来递推（节约空间）.
在每个状态中，l=j-i 越来越小，所以从 l=0 递推到 l=n
即：for k:=1 to c do for l:=0 to n do　for i:=1 to n do
　　递推 d[i,j,k]
显然，用两个数组记录 d[i,j,k]和 d[i,j,k-1],就可以只用两维：d[i,j]
于是算法框架就是(请对照我的源程序)：
初始化 d1[i,j]
for k:=1 to c do
begin
  for l:=0 to n do
　for i:=1 to n do
　　用 d1[i,j](k-1 阶段)递推 d2[i,j]（k 阶段）

```
  d1:=d2; {节省空间，因为递推的时候只与上个阶段有关，故只保留相邻两个阶段}
end;
```

高精度乘法的方法我不是用的模拟手算的过程（这个大家会做吧），而用了类似多项式乘法的方法，因为我觉得这种写法很好记！程序见下。

```
const  maxn=50;  ch:string[20]='0123456789ABCDEFGHIJ';

var  i,j,n,m,k:longint;  bit:array[0..maxn] of longint;

begin  while not eof(input) do  begin  readln(n,m);  i:=0; k:=n;

    while (k<0)or(k>=-m) do

    begin  bit[i]:=k mod m;  k:=k div m;

      if bit[i]<0 then  begin  bit[i]:=bit[i]-m;  k:=k+1;  end;

      inc(i);  end;

    bit[i]:=k;

    for j:=i downto 0 do write(ch[bit[j]+1]);  writeln;

  end;

end.
```

题三：单词接龙（27 分）

问题描述：单词接龙是一个与我们经常玩的成语接龙相类似的游戏，现在我们已知一组单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次），在两个单词相连时，其重合部分合为一部分，例如 beast 和 astonish，如果接成一条龙则变为 beastonish，另外相邻的两部分不能存在包含关系，例如 at 和 atide 间不能本连。

输入：输入的第一行为一个单独的整数 N(N<=20)表示单词数，以下 N 行每行有一个单词，输入的最后一行为一个单个字符，表示“龙”开头的字母。你可以假定以此开头的“龙”一定存在。

输出：只需输出以此字母开头的最长的“龙”的长度

样例：输入：5

at

touch

cheat

choose

tact

a

输出：23 （连成的“龙”为 atoucheatactactctouchoose）

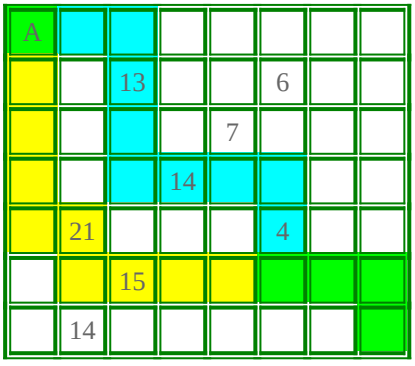
分析：搜索。数据很小，因此回溯就可以了。程序先预处理,建立矩阵 add[i,j]，即第 j 个串连在第 i 个串之后能增加的长度。0 代表 j 不能增加 i 的后面。
一个需要注意的地方：计算 add[i,j]的时候要计算最大可能值！
例如：ABABABAB 和 ABABABC 就是 5,不是 1！
现在没有问题了吧。为了方便和直观，我采用递归实现回溯搜索。程序见下。

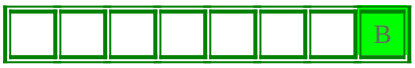
```
const  maxn=20;

var  s:array[1..maxn] of string;  head:char;  best,i,n:integer;

  add:array[1..maxn,1..maxn] of integer;

  used:array[1..maxn] of integer;

procedure calcadd;

var  i,j,k,t,min:integer;  ok:boolean;

begin  for i:=1 to n do   for j:=1 to n do   begin

  if length(s[i])<length(s[j]) then min:=length(s[i])  else min:=length(s[j]);

      for k:=1 to min-1 do   begin    {check}

        ok:=true;

        for t:=1 to k do  if s[j,t]<>s[i,length(s[i])-k+t] then

          begin  ok:=false;   break;   end;

        if ok then break;

      end;

      if ok then  add[i,j]:=length(s[j])-k

      else  add[i,j]:=0;  end;
```

```
end;

procedure search(last,len:integer);

var  i:integer;

begin  if len>best then  best:=len;

  for i:=1 to n do if (add[last,i]>0)and(used[i]<2) then

    begin  inc(used[i]);  search(i,len+add[last,i]);  dec(used[i]);

    end;

end;

begin  readln(n);  for i:=1 to n do  readln(s[i]);  readln(head);

  calcadd;  best:=0;  fillchar(used,sizeof(used),0);

  for i:=1 to n do  if s[i,1]=head then

    begin  used[i]:=1;   search(i,length(s[i]));   used[i]:=0;

    end;  writeln(best);

end.
```

题四：方格取数（33 分）

问题描述：设有 N*N 的方格图(N<=8)，我们将其中的某些方格中填入正整数，则其他的方格中则放入数字 0，如下图所示设有 N*N 的方格图(N<=8)，我们将其中的某些方格中填入正整数，而其他的方格中则放人数字 0。如下图所示（见样例 ，黄色和蓝色分别为两次走的路线，其中绿色的格子为黄色和蓝色共同走过的）：





某人从图的左上角的 A 点出发，可以向下行走，也可以向右行走，直到到达右下角的 B 点。在行走的路上，他可以取走方格中的数（取走后的方格中的数将变为数字 0）。

此人从 A 点到 B 点共走两次，试找出两条这样的路径，使得取得的数之和最大。

输入：输入的第一行为一个整数 N（表示 N*N 的方格图），接下来的每行有三个整数，前两个表示位置，第三个为该位置上所放的数。一行单独的 0 表示结束。

输出：只需输出一个整数，表示 2 条路径上取得的最大的和。

样例：输入：8　　5 2 21

2 3 13　　5 6 4

2 6 6　　6 3 15

3 5 7　　7 2 14

4 4 14　　0 0 0

输出：67

分析：有同学搜索第一个人，拣了以后第二个人用动态规划，一定能得最优解，但时间效率不大高。有同学采用贪心，即用动态规划算出第一个人最大能拣的数，再在剩下的数中用动态规划。
反例如下：
1 9 1
0 0 0
1 9 1
第一次是：1->9->9->1　第二次是：1　　和是 21
但显然可以两次把所有的数拣完(22)。
本题是典型的多维动态规划，很象 IOI93 的第四题,另一个算法是网络流，很象 IOI97 第一题，这里我只分析前者。这道题目的简单之处是阶段很好划分（对角线），这种方法我就不介绍了，因为很多地方都有介绍。这里讲一种怪一点的动态规划^_^
容易想到的思路是：令 d[x1,y1,x2,y2]表示甲在 x1,y1,乙在 x2,y2 以后（包括取走(x1,y1)）的过程中可以获得的最大和。则 d[1,1,1,1]就是原问题的解。
但是是否能取数和另一个人是否事先已经到过该格子有关，我们需要设计一种走的方法，使得只根据 x1,y1,x2,y2 就能判断一些关键的格子是否已经到达过。这样，问题才具有无后效性。
为此，我们把路线作如下处理：1)当两个人路线有交叉的时候，改成等效的，不交叉的。如下图，1 代表第一个人，2 代表第二个人。X 代表相遇点。
X111
2　1
222X2

12
12
1X1
2X
变成：
X222
1 2
111X2
12
12
1X2
1X
反正让 2 走右边就行了。
2)现在 1 在第 y1 列，2 在第 y2 列，让 1 和 2 分别向右走,到达 yy1 和 yy2 列，然后向下走一格，这样如果 yy1 < y2,便是分别取走第 y1~yy1,y2~yy2 列数，否则路线有重复，就取走 y1~yy2 的数。为了方便连续取数，我用了一个 sum[x,y1,y2]的数组，就是第 x 行的 y1~y2 的数。请看我的程序中的相应部分。
这样，所有的走法都可以转换成上述的，具有无后效性的结构了。
由于递推是从 d[x1,y1,x2,y2]到 d[x1+1,y1',x2+1,y2'],而总有 x1=x2=x,所以可以把状态节省为：d[y1,y2]，而把 x(当前行)作为阶段来递推：

```
for x:=n-1 downto 1 do
begin
  for y1:=1 to n do
    for y2:=y1 to n do
      枚举 y1'和 y2'作为新的 y1 和 y2,注意保证 y1' >= y1, y2' >= y2(题目规定), y1' <= y2'(刚才的分析),递推 d[y1,y2]
  d1:=d2; {只记录相邻两个状态}
end;
```

边界是什么呢？当然是从第 n 列的值了，就是 sum[n,y1,n](从 y1 取到 n)
说了这么多，真不知道我说清楚了没有（ 好象是没有:-P ），如果有什么不明确的地方，请大家在论坛上提出来。一句话，就是两个人一起，一行一行往下走，每一行可以一次向右走若干步，但是要保证 2 在 1 的右边。
注意最后输出的是 d2[1,1]。如果输出 d1[1,1],n=1 会得到 0。（为什么？自己想啊）
现在程序就不难写了吧。程序见下：

```
const  maxn=10;

var  n:integer;

  m:array[1..maxn,1..maxn] of integer;

  d1,d2:array[1..maxn,1..maxn] of integer;

  sum:array[1..maxn,1..maxn,1..maxn] of integer;

procedure init;
```

```
var  x,y,p:integer;  i,j,k:integer;

begin  readln(n);  fillchar(m,sizeof(m),0);

  repeat   readln(x,y,p);

    if (x=0)and(y=0)and(p=0) then break;   m[x,y]:=p;

  until false;         {calc sum}

  for i:=1 to n do  begin  sum[i,1,1]:=m[i,1];

    for j:=2 to n do   sum[i,1,j]:=sum[i,1,j-1]+m[i,j];

    for j:=2 to n do   for k:=j to n do

        sum[i,j,k]:=sum[i,1,k]-sum[i,1,j-1];

  end;

end;

function max(a,b:integer):integer;

begin if a>b then max:=a else max:=b; end;

procedure solve;

var  y1,y2,yy1,yy2,x,r:integer;

begin  {init}

  for y1:=1 to n do   for y2:=y1 to n do   d2[y1,y2]:=sum[n,y1,n];

  for x:=n-1 downto 1 do

  begin   for y1:=1 to n do   for y2:=y1 to n do

      begin   d1[y1,y2]:=-maxint;

        for yy1:=y1 to n do   for yy2:=max(y2,yy1) to n do

          begin  if yy1>=y2 then r:=sum[x,y1,yy2]+d2[yy1,yy2]

              else r:=sum[x,y1,yy1]+sum[x,y2,yy2]+d2[yy1,yy2];
```

```
            if r>d1[y1,y2] then d1[y1,y2]:=r;
          end;
      end;    d2:=d1;
  end;
end;
begin  init;  solve;  writeln(d2[1,1]);   end.
```

附：解题报告及源程序(刘汝佳)：http://www.shzx.net.cn/cms/oi/shiti/NOIP2000T_report_liurujia.zip

附：源程序(袁豪)：http://www.shzx.net.cn/cms/oi/shiti/NOIP2000T_report_yuanhao.zip

附：测试数据：http://www.shzx.net.cn/cms/oi/shiti/2000fstdata.zip