第五届(1999 年)全国青少年信息学(计算机)奥林匹克分区联赛

(NOIP)提高组复赛试题（三小时完成）

第一题　拦截导弹（20 分）

某国为了防御敌国的导弹袭击, 发展出一种导弹拦截系统. 但该系统有一个缺陷:虽然它的第一发炮弹能够到达任意高度, 但是以后每一发炮弹都不能高于前一发的高度.某天, 雷达捕捉到敌国的导弹来袭, 由于该系统还在试用阶段, 所以只有一套系统, 因此有可能不能拦截所有的导弹.

输入导弹依次飞来的高度(雷达给出的高度数据是不大于 30000 的正整数), 计算这套系统最多能拦截多少导弹, 如果要拦截所有导弹最少要配备多少套这种导弹拦截系统.

样例: INPUT　　389 207 155 300 299 170 158 65

OUTPUT　6(最多能拦截的导弹数)

2(要拦截所有导弹最少要配备的系统数)

分析: 第一部分是要求输入数据串中的一个最长不上升序列的长度, 可使用递推的方法. 具体做法是从序列的第一个元素开始, 依次求出第 I 个元素为最后一个元素时的最长不上升序列的长度 Len(I), 递推公式为 Len(1)=1, Len(I)=Max(Len(J)+1), 其中 I>1, J=1,2,...,I-1, 且 J 同时要满足条件: 序列中第 J 个元素大于第 I 个元素. 例如输入数据为 75 4 1 6 3 2, 可算得 Len(1)=1, Len(2)=2, Len(3)=3, Len(4)=4, Len(5)=2, Len(6)=4,Len(7)=5, 所以最长不上升序列是 5, 即 7 5 4 3 2.

第二部分比较有意思, 由于它紧接着上一问, 所以比较时很容易受前面的影响, 采取多次求最长不上升序列的办法, 然后得出总次数, 其实这是不对的. 要举反例并不难, 比如对序列 7 5 4 1 6 3 2, 用多次最长不上升序列的结果为 3 套系统, 但其实只要 2 套, 分别击落 7 5 4 1 与 6 3 2.

所以正确的做法不应受"一套系统尽量多拦截导弹"的思维定势的影响, 应该换一个思路, 从拦截其个导弹所选的系统入手. 如对于第 I 个数据, 如果已有系统能拦截, 即数据不大于前面系统的最后一个数据时, 就用原有系统拦截, 否则启用一个新系统. 如果已有系统有多个系统可拦截时, 选用哪一个呢? 当用"潜力"最大者, 即各系统中最后数据较小者取之.

```
Const Max=1000;

Var I,J,Current,Maxlong,Minheight,Select,Tail,Total:Longint;

   Height,Longest,Sys:Array[1..Max] Of Longint;

   Line:String;
```

```
Begin  {Input And String->Val}    {输入数据串并转为数值存放于数组}
 Write('Input Test Data:'); Read(Line); I:=1;
 While I<=Length(Line) Do Begin
  While (I<=Length(Line)) And (Line[I]=' ') Do I:=I+1;  {跳过串空格}
  Current:=0;    While (I<=Length(Line)) And (Line[I]<>' ') Do
    Begin Current:=Current*10+Ord(Line[I])-Ord('0');  I:=I+1;
      End;  {Total 数据个数}
  Total:=Total+1; Height[Total]:=Current End;  {数据存于 Height 数组}
 {Compute Step }                  {计算最多拦截导弹数}
 Longest[1]:=1; ForI:=2 To Total Do Begin Maxlong:=1;  {Len(1)=1}
  For J:=1 To I-1 Do Begin
       If (Height[I]<=Height[J]) And (Longest[J]+1>Maxlong)
      Then Maxlong:=Longest[J]+1;            {Len(I)=Len(J)+1}
  Longest[I]:=Maxlong End; End;    Maxlong:=Longest[1];
 For I:=2 To Total Do If Longest[I]>Maxlong
          Then Maxlong:=Longest[I]; {找个数最大者}
  Writeln('Max Step=',Maxlong);
 {Compute System}              {计算最少配备系统数}
 Sys[1]:=Height[1]; Tail:=1;           {Tail 系统数}
 For I:=2 To Total Do Begin   Minheight:=Maxint;
  For J:=1 To Tail Do   If (Sys[J]>Height[I]) And (Sys[J]<Minheight)
          Then Begin Minheight:=Sys[J]; Select:=J End;
  If   Minheight=Maxint   Then
```

```
        Begin  Tail:=Tail+1; Sys[Tail]:=Height[I] End  {启用新系统}

    Else Sys[Select]:=Height[I] End;          {使用已有系统}

   Writeln('Min System=',Tail)        End.
```

第二题　回文数（25 分）

[问题描述] 若一个数(首位不为 0)从左向右读与从右向左读都是一样, 我们就将其称之为回文数。例如: 给定一个 10 进制数 56, 将 56+65(即把 56 从左向右读), 得到 121 是一个回文数。

又如: 对于 10 进制数 87:

STEP1: 87+78=165　　STEP2: 165+561=726

SET3: 726+627=1353　　STEP2: 1353+3531=4884

在这里的一步是指进行了一次 N 进制的加法, 上例最少用了 4 步得到回文数 4884.

写一程序, 给定一个 N(2<=N<=10 或 N=16)进制数 M, 求最少经过几步可得到回文数.

如果在 30 步以内(包含 30 步)不可能得到回文数, 则输出"Impossible!"

样例: INPUT　　OUTPUT

　　N=9　M=87　　STEP=6

分析: 因本题数据的进制是可变的, 所以用整型数不方便, 采用字符串读入, 再把它的每一位分离出来存入一个数组里, 数组的 0 下标记录数据位数. 在转化过程中要注意 16 进制的特殊性, 对于 ABCDEF 要单独处理.

运算要符合 N 进制的规律, 这里一律采用 10 进制数存储, A 用 10, F 用 15. 判断是否回文数时也采用 10 进制, 逐对比较对称的两个数位上的数字, 看它们是否相等, 做加法的次数以 30 为限.

题目要求输出内容为: 是回文数则输出该数, 不是则输出提示. 下面的程序除了要求的输出内容外, 还输出了每步的运算情况, 为了简便可以省去.

```
Const Max=100;

  Setofchar=['A','B','C','D','E','F'];   Setofchar0=['a','b','c','d','e','f'];

Type Tdata=Array[0..Max] Of Integer;
```

```
Var D:Tdata;  I,N,Step:Integer; Source:String;
Function Check(A:Tdata):Boolean;                {检测是否回文数}
  Var I,J:Integer;  Temp:Boolean;
  Begin Temp:=True; J:=A[0]; I:=1;
  While Temp And (I<J) Do Begin If A[I]<>A[J] Then Temp:=False;
    Inc(I); Dec(J); End;   Check:=Temp; End;
Procedure Add(Var A:Tdata;B:Tdata);     Var I:Integer;     {两数相加}
  Begin For I:=1 To D[0] Do
    Begin Inc(A[I],B[I]); Inc(A[I+1],A[I] Div N); A[I]:=A[I] Mod N  End;
  If A[A[0]+1]>0 Then Inc(A[0]);  End;
Procedure Solve;      Var I:Integer; T:Tdata;       {处理回文数}
  Begin Step:=0;
  While (Step<30) And Not Check(D) Do Begin  Inc(Step); Write(Step:2,': ');
    For I:=1 To D[0] Do T[I]:=D[D[0]+1-I]; T[0]:=D[0];   {取反向数}
      If Step=1 Then Begin
        For I:=1 To D[0] Do Write(D[I]); Write('+');
          For I:=1 To T[0] Do Write(T[I]); Write('='); End
              Else Begin {划线部分是副输出}
      For I:=1 To T[0] Do Write(T[I]); Write('+');          .
      For I:=1 To D[0] Do Write(D[I]); Write('='); End;
    Add(D,T);     For I:=D[0] Downto 1 Do Write(D[I]); Writeln;
    End;  End;
Begin Write('N(2..10 Or 16)=');  Readln(N);             {主程序,输入}
```

```
  Write('Number=');  Readln(Source);
  While Pos(' ',Source)<>0 Do Delete(Source,Pos(' ',Source),1); {掉串中空格}
  D[0]:=Length(Source);                    {串长}
  For I:=1 To D[0] Do                      {串转数}
    If Source[I] In Setofchar Then D[I]:=Ord(Source[I])-Ord('A')+10
    Else If Source[I] In Setofchar0 Then D[I]:=Ord(Source[I])-Ord('a')+10
    Else D[I]:=Ord(Source[I])-Ord('0');          Solve;
  If Check(D) Then Writeln('Step=',Step) Else Writeln('Impossible!');
End.
```

第三题　旅行家的预算（27 分）

一个旅行家想驾驶汽车人最少的费用从一个城市到另一个城市（假设出发时油箱是空的）。给定两个城市之间的距离 D1，汽车油箱的容量 C（以升为单位），每升汽油能行驶的距离 D2，出发点每升汽油价 P 和沿途汕站数 N（N 可以为 0），油站　i 离出发点的距离 Di ，每升汽油价格 Pi (i=1,2,...n)。

计算结果四舍五入到小数点后两位。

如果无法到达目的地，则输出“No Solution”

样例：INPUT　D1=275.6　C=11.9　D2=27.4　P=2.8　N=2

油站号 i	离出发点的距离 Di	每升汽油价格 Pi
1	102.0	29
2	220.0	2.2

OUTPUT　26.95 （该数据表示最小费用）

第四题　邮票面值设计（40 分）

给定一个信封，最多只允许粘贴 N 张邮票，计算在给不定期 K(K+N<=40)种邮票的情况下(假定所有的邮票数量都足够)，如何设定邮票的面值，能得到最大值 MAX，使在 1--MAX 之间的每一个邮资值都能得到。

假如，N=3，K=2，如果面值分别为 1 分、4 分，则在 1 分--6 分之间的每一个邮资都能得到(当做还有 8 分、9 分和 12 分)；如果面值分别为 1 分、3 分，则在 1 分--7 分之间有每

一个邮资值都能得到。可以验证当 N=3，K=2 时，7 分就是可以得到的连续的邮资最大值，所以 MAX=7，面值分别为 1 分、3 分。

样例：INPUT　　OUTPUT

N=3　K=2　　1　3

MAX=7

分析：考试的时候给出的数据规模是 N+K<=40，因此不少选手都想方设法采用递推；实际上，本题具有明显的后效性，前面方案的不同，使得构成某面值的邮票数不同，递推是不成立的。一般认为解决这类问题的方法只有递归搜索，所能解决的问题规模也远远达不到题目要求(N 和 K 基本不能同时超过 6)，这可以说是条件中的一个陷阱。即使对于较小的 N 和 K，要想出解也要具有正确的思路才行。

面值为 1 的邮票是必须的，可以固定，以后每加进一种面值都要把当前所能取得的金额记录下来，通过 K-1 次的添加得到一种方案，穷举所有方案得到最优解，这就是算法的基本框架。

在搜索的过程中，对数据和记录方式很关键。如果仅仅记录一个金额是否能达到，则这样的记录基本上没有用，因为下一次添加邮票时不知道能否从这个金额出发再加上一张新邮票，必须全部重算，浪费了大量时间；所以记录用几张邮票达到一个面值是比较方便的，它有利于添加新邮票时迅速判断可行性。另外，搜索一种邮票面值的起讫点也要注意应当从上一个面值加 1 直到当前连续取得的最大面值加 1。在这个范围之外的解可以不予考虑：过大导致不连续；小于前一面值时可将前后面值交换，仍得到增序。

```
Program Noi99_4;

Const Max=250;  Type Ar=Array[0..Max] Of Longint;

Var I,Kind,Num,M:Longint; S,R:Array[1..10] Of Longint; D:Ar;

Function E(Var D:Ar):Longint;        Var I,C:Longint;

  Begin C:=1; While D[C]<=Num Do Inc(C); E:=C; End;

Procedure Try(Dep,Maxc:Longint; D:Ar);

  Var I,J,K:Longint; Temp:Ar;

  Begin If Dep>Kind Then If Maxc>M Then Begin M:=Maxc; R:=S End

    Else Else For I:=S[Dep-1]+1 To Maxc Do Begin S[Dep]:=I; Temp:=D;

      For J:=1 To Num-1 Do For K:=0 To Max-J*I Do
```

```
        If Temp[K+J*I]>D[K]+J Then Temp[K+J*I]:=D[K]+J;
        If Num*I<=Max Then Temp[Num*I]:=Num;
        Try(Dep+1,E(Temp),Temp) End; End;
Begin Write('N=');Readln(Num); Write('K='); Readln(Kind);
  S[1]:=1; For I:=1 To Max Do D[I]:=Maxint;
  For I:=1 To Num Do D[I]:=I; D[0]:=0; M:=0; Try(2,E(D),D);
  For I:=1 To Kind Do Write(R[I],' '); Writeln;
  Writeln('Max=',M-1); End.
```

附：解题报告、源程序（刘汝佳）：http://www.shzx.net.cn/cms/oi/shiti/NOIP1999T_report_liurujia.zip

附：源程序（袁豪）：http://www.shzx.net.cn/cms/oi/shiti/NOIP1999T_report_yuanhao.zip

附：测试数据：http://www.shzx.net.cn/cms/oi/shiti/1999fstdata.zip