CÁC THUẬT TOÁN SỐ HỌC

I. SỐ NGUYÊN TỐ

1. Đinh nghĩa:

Một số tự nhiên p (p > 1) là số nguyên tố nếu p có đúng hai ước số là 1 và p.

Ví dụ các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23, 29…

2. Kiểm tra tính nguyên tố

Để kiểm tra một số nguyên dương n (n > 1) có là số nguyên tố hay không, ta kiểm tra xem có tôn ftại một số nguyên k ($2 \le k \le n-1$) mà k là ước của n (n chia hết cho k) thì n không phải là số nguyên tố, ngược lại n là số nguyên tố.

Tuy nhiên, nếu n (n > 1) không phải là số nguyên tố, ta luôn có thể tách $n = k_1 \times k_2$ mà $2 \le k_1 \le k_2 \le n-1$. Vì $k_1 \le k_2$ nên $k_1 \times k_1 \le k_1 \times k_2 = n \Rightarrow k_1^2 \le n$

hay $k_1 \le \sqrt{n}$. Do đó việc kiểm tra từ 2 đến n - 1 là không cần thiết, mà ta chỉ cần kiểm tra k từ 2 đến $\sqrt{n}$.

<u>Bài 1:</u> Nhập vào một số nguyên dương, kiểm tra xem nó có phải là số nguyên tố hay không?

Lời giải:

```
Program SO_NGUYEN_TO;
Uses crt;
Var i, n: integer;
Begin
    Clrscr;
    Writeln('KIEM TRA SO NGUYEN TO:');
    Write ('Nhap so can kiem tra n = ');   readln(n);
    If (n=0) or (n=1) then Writeln(n,'  khong phai la so
nguyen to')
    Else
        Begin
            i:=1;
            Repeat
                i:= i+1;
```

```
            Until (n mod i= 0) or (i>sqrt(n));
          If  i>sqrt(n)  then Writeln (n,' la so nguyen to')
          Else Writeln (n,'  khong phai la so nguyen to');
          End;
Readln;
End.
```

Nhận xét:

Chương trình trên tiến hành kiểm tra lần lượt từng số nguyên k trong đoạn $\left[2,\sqrt{n}\right]$

Để cải tiến cần giảm thiểu số các số cần kiểm tra. Ta có nhận xét, để kiểm tra một số nguyên dương n (n > 1) có là số nguyên tố không, ta kiểm tra xem có tồn tại một số nguyên tố k ($2 \le k \le \sqrt{n}$) mà k là ước của n thì n không phải là số nguyên tố, còn ngược lại thì n là số nguyên tố. Thay vì kiểm tra các số k là số nguyên tố ta sẽ chỉ kiểm tra các số k có tính chất giống với tính chất của số nguyên tố, có thể sử dụng một trong hai tính chất đơn giản sau của số nguyên tố:

1, Trừ số 2 các số nguyên tố là số lẻ.

2, Trừ số 2, số 3 các số nguyên tố có dạng $6k \pm 1$(vì số có dạng $6k \pm 2$ thì chia hết cho 2, số có dạng $6k \pm 3$ thì chia hết cho 3)

3. Liệt kê các số nguyên tố trong đoạn $\left[1,N\right]$

Phương pháp: Ta thử lần lượt các số m trong đoạn $\left[1,N\right]$**,** rồi kiểm tra tính nguyên tố của m.

<u>Bài 2</u>: In ra các số nguyên tố nhỏ hơn hoặc bằng N (N là số nguyên không âm được nhập từ bàn phím).

```
Program CAC_SO_NGUYEN_TO;
Uses crt;
Var N, i, t: integer;
Begin
    Clrscr;
    Writeln('IN RA CAC SO NGUYEN TO <=N');
    Write('Nhap N= ');  readln(N);
    If  N<2  then Writeln('Khong co so nguyen to nao <=',
N)
```

```
    Else
        Begin
            Writeln('Cac so nguyen to <= ', N,' la:');
            For i := 2 to N do
            Begin
                t:= 1;
                Repeat
                t:= t+1;
                Until ( i mod t = 0) or ( t>sqrt (i) ) ;
                If  ( t > sqrt(i))  then Write(i:4);
            End;
        End;
Readln;
End.
```

Cách thứ hai là sử dụng sàng số nguyên tố, như sàng Eratosthene, liệt kê được các số nguyên tố nhanh, tuy nhiên nhược điểm của cách này là tốn nhiều bộ nhớ. Cách làm được thực hiện như sau:

Trước tiên xóa bỏ số 1 ra khỏi tập các số nguyên tố. Số tiếp theo số 1 là số 2, là số nguyên tố, xóa tất cả các bội của 2 ra khỏi bảng. Số đầu tiên không bị xóa sau số 2 (số 3) là số nguyên tố, xóa các bội của 3…Giải thuật tiếp tục cho đến khi gặp số nguyên tố lớn hơn $\sqrt{N}$ thì dừng lại. Tất cả các số chưa bị xóa là số nguyên tố.

Thuật toán cụ thể như sau:

```
program lietke;
uses crt;
var n: longint;
procedure sang_nt(N: longint);
var
    i,j: longint;
    prime: array[ 1..100] of longint;
begin
     fillchar(prime,sizeof(prime),0);
      for i:=2 to trunc(sqrt(N)) do
        if prime[ i] =0 then
        begin
```

```
            j:=i*i;
            while j<=N do
               begin
                   prime[j]:=1;
                   j:=j+1;
               end;
         end;
         for i:=2 to N do
           if prime[i]=0 then writeln(i);
end;
begin
  clrscr;
  write('Nhap n= '); readln(n);
  sang_nt(n);
  readln;
end.
```

Bài 3. Bài toán số nguyên tố tương đương

Hai số tự nhiên được gọi là nguyên tố tương đương nếu chúng có chung các ước số nguyên tố. Ví dụ như các số 75 và 15 là nguyên tố tương đương vì cùng có các ước nguyên tố là 3 và 5.

Cho trước hai số tự nhiên M và N. Hãy viết chương trình kiểm tra xem các số này có là nguyên tố tương đương với nhau không?

```
Program Nttd;
Var M, N, d, i: Integer;
Function USCLN(M, N: integer): integer;
Var r: integer;
Begin
   While (N<>0) do
        Begin
                r:= M mod N;
                M:=N;
                N:=r;
        End;
USCLN:= M;
End;
```

```
    Begin
         Clrscr;
         Write('Nhap M, N='); readln(M, N);
         d:= USCLN(M, N);
         i:=2;
         while d<>1 do
               begin
                   if d mod i = 0 then
                       begin
                           while d mod i = 0 do d:=d div i;
                           while d mod i = 0 do d:=d div i;
                           while d mod i = 0 do d:=d div i;
                       end;
                   inc(i);
               end;
if M*N=1 then write(M, 'va', N, 'la hai so nguyen to tuong
duong')
else write(M, 'va', N, 'khong la hai so nguyen to tuong
duong');
    readln;
    end.
```

4. Số chính phương:

Trước hết, chúng ta sẽ tìm hiểu khái niệm về số chính phương. Số chính phương là gì? Số chính phương là một số mà tự nó là căn bậc hai của một số tự nhiên khác, hay nói rõ hơn thì số chính phương là bình phương của một số tự nhiên.

Ví dụ: 289 là một số chính phương vì 289 = 17 bình phương.

Thuật toán Pascal dưới đây sẽ giúp tìm số chính phương trong mảng 1 chiều.

```
  uses crt;
  type ArrInt = array[1..250] of integer;
  Var n,i,x : integer;
         a: ArrInt;
  BEGIN
   clrscr;
   write('Nhap so phan tu: ');
   readln(n);
   for i:=1 to n do
   begin
```

```
    write('Phan tu thu ',i,'= ');
    readln(a[i]);
  end;
  writeln('Cac so chinh phuong co trong mang:');
  for i:=1 to n do
  begin
   x:=trunc(sqrt(a[i]));
    if sqr(x)=a[i] then
      write(a[i]:4);
  end;
  readln;
END.
```

Trong đó lệnh hàm sqrt để lấy căn và hàm trunc để lấy phần nguyên.

II. ƯỚC SỐ, BỘI SỐ

1. Số các ước của một số

Giả sử N được phân tích thành thừa số nguyên tố như sau:

$$N = a^i \times b^j \times ... \times c^k$$

Khi đó ước số của N có dạng: $a^p \times b^q \times ... \times c^r$ trong đó:

$$0 \le p \le i, 0 \le q \le j, ..., 0 \le r \le k.$$

Do đó số các ước số của N là: $(i+1)\times(j+1)\times...\times(k+1)$.

Ví dụ:

$N = 100 = 2^2 \times 5^2$, số ước số của 100 là: $(2+1)(2+1) = 9$ ước số (Các ước số đó là: 1, 2, 4, 5, 10, 20, 25, 50, 100).

$N = 24 = 2^3 \times 3$, số ước số của 24 là: $(3+1)(1+1) = 8$ ước số (các ước số đó là: 1, 2, 3, 4, 6, 8, 12, 24).

2. Tổng các ước số của một số

$$N = a^i \times b^j \times ... \times c^k$$

Đặt $N1 = b^j \times ... \times c^k$

Gọi F(t) là tổng các ước của t, ta có:

$$F(N) = F(N1) + a \times F(N1) + ... + a^i \times F(N1)$$

$$= (1 + a + ... + a^i) \times F(N1) = \frac{(a^{i+1} - 1)}{a - 1} \times F(N1)$$

$$= \frac{(a^{i+1} - 1)}{a - 1} \times \frac{(b^{j+1} - 1)}{b - 1} \times ... \times \frac{(c^{k+1} - 1)}{c - 1}$$

Ví dụ: Tổng các ước của 24 là:

$$\frac{\left(2^{3+1}-1\right)}{2-1}\times\frac{\left(3^{1+1}-1\right)}{3-1}=60$$

Bài 4: Cho số nguyên dương N (N≤10^9)

a, Phân tích N thành thừa số nguyên tố

b, Đếm số ước của N

c, Tính tổng các ước của N

Gợi ý:

a, Ý tưởng: Thuật toán phân tích một số ra thừa số nguyên tố tương tự như thuật toán kiểm tra số nguyên tố. Điểm khác ở đây là khi kiểm tra số nguyên tố ta phải lần lượt kiểm tra các số nhỏ hơn sqrt(n) (căn bậc hai của n) có phải là ước của n hay không, còn khi phân tích ta chỉ việc chia n cho các số nguyên bắt đầu từ số nguyên tố nhỏ nhất là 2. Khi không chia được nữa thì ta tăng số chia lên 1 đơn vị, quá trình phân tích kết thúc khi n bằng 1.

```
VAR i,n :INTEGER;
BEGIN
    Write ('Nhap n:');
    Readln(n);
    Write (n,'=');
    i:=2;
    REPEAT
        WHILE n MOD i <> 0 DO
            i:=i+1;
        Write(i);
        n:=n DIV i;
        IF n > 1 THEN
            write ('*');
    UNTIL n = 1;
    readln;
END.
```

b, Đếm số ước của n: Ta cho i chạy từ 1 đến n div 2, nếu n chia hết cho số nào thì ta tăng số ước lên 1. Lưu ý rằng: n cũng là ước của n, nên số ước phải cộng thêm 1

c, Để tính tổng các ước số của số n, ta cho i chạy từ 1 đến n div 2, nếu n chia hết cho số nào thì ta cộng số đó vào tổng.

Chương trình cụ thể của phần b và phần c như sau:

```
program bai1;
uses crt;
```

```
var i,n: longint;
Function  dem(n: longint): longint;
var count: longint;
begin
    count:=0;
    for i:=1 to n div 2 do
       if n mod i =0 then count:=count +1;
    dem:=count+1;
end;
Function tong(n: longint): longint;
var S: longint;
begin
    S:=0;
    for i:=1 to n div 2 do
       if n mod i =0 then S:=S +i;
    tong:=S+n;
end;
Begin
    clrscr;
    write('Nhap n='); readln(n);
    write('So cac uoc cua n la:',dem(n));
    writeln;
    write('tong cac uoc cua n la:', tong(n));
    readln;
    end.
```

Bài 4: Hai số m, n gọi là bạn của nhau nếu tổng các ước của m bằng n và ngược lại. Tìm tất cả các số là bạn của nhau và nhỏ hơn 10001.
Ý tưởng: Ta có 24 và 60 là bạn của nhau vì tổng các ước của 24 bằng 60.

Thay vì chạy 2 vòng lặp để xét m và n, ta có thể chỉ cần chạy 1 vòng lặp kiểm tra xem m và uoc(m) có là bạn của nhau không.

```
PROGRAM timban;
FUNCTION uoc(k:INTEGER):longint;
VAR i,tong:INTEGER;
BEGIN
tong:=0;
```

```
FOR i:=1 TO k DIV 2 DO
IF k MOD i =0 THEN tong:=tong+i;
uoc:=tong;
END;
VAR m:longint;
BEGIN
for m:= 1 to 10001 do
   if uoc(uoc(m)) = m then writeln(m, ' va ', uoc(m),' la
ban cua nhau');
readln
END.
```

3. Ước số chung lớn nhất của hai số

Ước số chung lớn nhất (USCLN) của hai số được tính theo thuật toán Euclid

USCLN(a,b)=USCLN(b,(a mod b))

4. Bội số chung nhỏ nhất của hai số

Bội số chung nhỏ nhất (BSCNN) của hai số được tính theo công thức:

$$BSCNN(a,b) = \frac{a \times b}{\mathrm{US}CLN(a,b)}$$

Bài 5: Least Common Multiple

Bội số chung nhỏ nhất của một tập các số nguyên dương là số nguyên dương nhỏ nhất mà nó chia hết cho tất cả các số trong tập đó. Ví dụ, bội số chung nhỏ nhất của 5, 7 và 15 là 105.

Hãy viết một chương trình tìm bội số chung nhỏ nhất của một tập gồm n số nguyên dương cho trước.

Dữ liệu: File vào gồm hai dòng. Dòng đầu tiên chứa số nguyên dương n ($1 \le n \le 22$) và dòng thứ hai chứa n số nguyên dương a_1, a_2, ..., a_n ngăn cách nhau bởi một dấu cách, tất cả các số có giá trị nằm trong khoảng của số nguyên 32-bit.

Kết quả: File ra gồm một dòng chứa một số là bội số chung nhỏ nhất của n số a_1, a_2, ..., a_n. Giả thiết rằng kết quả nằm trong khoảng số nguyên 32-bit.

Ví dụ:

lcm.inp	lcm.out
3 5 7 15	105
6 4 10296 936 1287 792 1	10296

```
const fi='D:\lcm.inp';
```

```
      fo='D:\lcm.out';
var i,j,m,n: longint;
     a: array[1..22]  of longint;
     f,g:text;
Procedure doc;
Begin
     assign(f,fi);
     reset(f);
     readln(f,n);
     for i:= 1 to n do   read(f,a[i]);
     close(f);
end;
function UCLN(x,y: longint): longint;
       var sodu: longint;
Begin
      while y<>0 do
          Begin
              sodu:=x mod y;
              x:=y;
              y:=sodu;
          end;
      UCLN:=x;
end;
Function BCNN(x,y: longint):longint;
Begin
     BCNN:= (x*y) div UCLN(x,y);
end;
Procedure ghi;
Begin
   assign(g,fo);
   rewrite(g);
   m:=1;
   for i:= 1 to n do
       m:=BCNN(m,a[i]);
   write(g,m);
   close(g);
end;
Begin
```

```
   doc;
   ghi;
end.
```

Bài 6: Cho N là một số nguyên dương không vượt quá 10^9. Hãy tìm số chữ số 0 tận cùng của N!

Hướng dẫn:

Ý tưởng cách tìm: Xét tất cả các số chia hết cho 5. Giả sử mỗi số đó có thể chia hết cho Xi chữ số 5. Cộng tất cả các Xi đó lại thì ta được số chữ số 0. Giả sử 25! = 15511210043330985984000000 có 6 chữ số 0 tận cùng.

Ta có:

5 chia hết cho 1 chữ số 5

10 chia hết cho 1 chữ số 5

15 chia hết cho 1 chữ số 5

20 chia hết cho 1 chữ số 5

25 chia hết cho 2 chữ số 5

-> suy ra tổng là 6 (đúng với kết quả là có 6 chữ số 0).

Chương trình cụ thể như sau:

```
var
n, i, j, count: longint;
begin
write('Nhap N (N>=1): '); readln(n);
for i:=1 to n do
begin
j:=i;
while j mod 5 = 0 do
begin
j:=j div 5;
count:=count+1;
end;
end;
write(' So chu so 0 cuoi cua ',n,'! la: ',count); readln;
end.
```

III. Số Fibonacci

Số Fibonacci được xác định bởi công thức sau:

$$\begin{cases} F_0 = 0 \\ F_1 = 1 \\ F_n = F_{n-1} + F_{n-2} \end{cases}$$

Số Fibonacci là đáp án của bài toán:

Bài toán cổ về việc sinh sản của các cặp thỏ như sau:

- Các con thỏ không bao giờ chết;
- Hai tháng sau khi ra đời, mỗi cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực, một cái);
- Khi đã sinh con rồi thì cứ mỗi tháng tiếp tho chúng lại sinh được một cặp con mới.

Giả sử từ đầu tháng 1 có một cặp mới ra đời thì đến giữa tháng thứ n sẽ có bao nhiêu cặp.

Ví dụ: n=5, ta thấy:

Giữa tháng thứ 1: có 1 cặp (chính là cặp ban đầu)

Giữa tháng thứ 2: có 1 cặp (vì cặp ban đầu vẫn chưa đẻ)

Giữa tháng thứ 3: 2 cặp (cặp ban đầu đẻ ra thêm một cặp con)

Giữa tháng thứ 4: 3 cặp (cặp ban đầu tiếp tục đẻ)

Giữa tháng thứ 5: 5 cặp (cặp ban đầu đẻ thêm một cặp và cặp sinh ra ở giữa tháng thứ 3 đẻ thêm một cặp).

Bài 7: Tính số Fibonacci thứ n bằng phương pháp lặp sử dụng công thức

$F_n = F_{n-1} + F_{n-2}$ với $n \geq 2$ và $F_0 = 0, F_1 = 1$

Chương trình cụ thể như sau:

```
program fibonacci;
uses crt;
var n: longint;
function Fibo(n:longint): longint;
var fi_1, fi_2,fi,i: longint;
Begin
     if n<=1 then exit;
     fi_2:=0; fi_1:=1;
     for i:=2 to n do
        begin
         fi:=fi_1+fi_2;
         fi_2:=fi_1;
         fi_1:=fi;
       end;
     write(fi);
end;
begin
   clrscr;
```

```
   write('Nhap n='); readln(n);
   fibo(n);
   readln;
end.
```

Bài 8: Hãy viết chương trình máy tính để nhập từ bàn phím số nguyên dương M (2<M<2000000000), rồi xuất ra màn hình số FIBONACI lớn nhất là nguyên tố và nhỏ hơn M.

Ví dụ: Với M=10 thì các số FIBONACI nhỏ hơn M là: 0, 1, 1, 2, 3, 5, 8. Số 5 là số nguyên tố lớn nhất trong các số FIBONACI nhỏ hơn M. Vậy cần đưa ra màn hình dòng thông báo kết quả: Số cần tìm là: 5.

```
uses crt;
var j,i,m,a,b,t:longint;
Function kt(n:longint):boolean;
 var i,d:integer;
  begin
  kt:=false;
  d:=0;
   For i:=1 to n do
    if n mod i=0 then inc(d);
  if d=2 then kt:=true;
 end;
begin
 clrscr;
 Write('Nhap m= ');
 readln(m);
 a:=0;
 b:=1;
  Repeat
   a:=a+b;
   b:=a+b;
  Until (a>=m) and (b>=m);
if a<b then begin t:=a;a:=b;b:=t;end;
Repeat
   a:=a-b;
   b:=b-a;
  Until ( (kt(a)) and (a<m))  or ( (kt(b)) and (b<m) );
If a>b then writeln(a);
 if b>a then writeln(b);
```

```
  readln
 end.
```

IV. Số Catalan

Số Catalan được xác định bởi công thức:

$$Catalan_n = \frac{1}{n+1}C_{2n}^{n} = \frac{(2n)!}{(n+1)!n!} \text{ với } n \geq 0$$

Một số phần tử đầu tiên của dãy Catalan là:

N	0	1	2	3	4	5	6	…..
$Catalan_n$	1	1	2	5	14	42	132	…..

Số Catalan là đáp án của các bài toán:

1) Có bao nhiêu cách khác nhau đặt n dấu ngoặc mở và n dấu ngoặc đóng đúng đắn?

Ví dụ: n=3 ta có 5 cách sau:

((( ))),(( )( )), (( ))( ), ( )(( )), ( )( )( )

2) Có bao nhiêu cây nhị phân có đúng (n+1) lá?

Bài 9: Tính số $Catalan_n$ (n≤100)

Hướng dẫn:

Đối với bài toán này để tính được số $Catalan_n$ ta cần áp dụng công thức:

$$Catalan_n = \frac{1}{n+1}C_{2n}^{n} = \frac{(2n)!}{(n+1)!n!}$$

Phần còn lại chỉ là bài toán tính giai thừa mà HS đã được học. Chương trình cụ thể như sau:

```
program catalan;
uses crt;
var n: longint;
    Ca:real;
function gt(n: longint): longint;
var S,i: longint;
Begin
    S:=1;
    for i:=1 to n do S:=S*i;
    gt:=S;
end;
Begin
   clrscr;
   write('Nhap n= '); readln(n);
   Ca:=gt(2*n)/(gt(n+1)*gt(n));
   write(Ca:6:2);
```

```
    readln;
end.
```

V. Một số bài tập khác:

Bài 1: John Smith quyết đinh đánh số trang cho quyển sách của anh ta từ 1 đến N. Hãy tính toán số lượng chữ số 0 cần dùng, số lượng chữ số 1 cần dùng,…, số lượng chữ số 9 cần dùng.

Dữ liệu vào trong file: DIGITS.INP gồm một dòng duy nhất chứa một số N (N≤10^100).

Kết quả ra file DIGITS.OUT có dạng gồm 10 dòng, dòng thứ nhất là số lượng chữ số 0 cần dùng, dòng thứ hai là số lượng chữ số 1 cần dùng, …, dòng thứ 10 là số lượng chữ số 9 cần dùng.

Ví dụ:

digits.in	digits.out
13	0 1 1 6 2 2 3 2 4 1 5 1 6 1 7 1 8 1 9 1

Gợi ý: Đối với bài toán này ta cần lưu ý như sau: Để xét chữ số tận cùng của số N ta xét N mod 10, giả sử N mod 10 = i thì số i sẽ xuất hiện một lần. Để xét chữ số trước chữ số tận cùng ta sẽ gán N:= N div 10 (tức là phần nguyên khi chia N cho 10), sau đó lại xét số tận cùng của số này. Làm tương tự như vậy cho đến khi N div 10 = 0.

Chương trình cụ thể của bài toán này như sau:

```
program digits;
uses crt;
const
    fi='C:\digits.inp';
    fo='C:\digits.out';
var
    n,i: longint;
    dem: array[0..9] of integer;
    f: text;
```

```
procedure doc;
Begin
   assign(f, fi); reset (f);
   readln(f,n);
   close(f);
end;
Procedure xu_ly;
var  so,i,j: longint;
begin
    fillchar(dem,sizeof(dem),0);
    for i:=1 to n do
        Begin
            j:=i;
            while j>0 do
               begin
                  so:=j mod 10;
                  dem[so]:=dem[so]+1;
                  j:=j div 10;
               end;
        end;
end;
Procedure ghi;
Begin
     assign(f,fo); rewrite(f);
     for i:=0 to 9 do  writeln (f,i,' ',dem[i]);
     close(f);
end;
begin
     doc;
     xu_ly;
     ghi;
end.
```

Bài 2: A Mathematical Curiosity

File vào MATH.INP

File ra MATH.OUT

File chương trình MATH.PAS

Giới hạn thời gian 1 giây

Cho trước hai số nguyên *n* và *m*, bạn hãy đếm số cặp số nguyên (*a*, *b*) sao cho $0 < a < b < n$ và $(a^2+b^2+m)/(ab)$ là một số nguyên.

Dữ liệu: File vào chứa nhiều trường hợp kiểm tra. Mỗi trường hợp kiểm tra được cho trên một dòng chứa hai số nguyên n và m. Kết thúc file vào là một trường hợp với n = m = 0.

Kết quả: Với mỗi trường hợp kiểm tra, ghi ra file ra số các cặp (a, b) thoả mãn yêu cầu bài toán. Mỗi kết quả ghi trên một dòng theo định dạng như ví dụ mẫu dưới đây.

Ví dụ:

MATH.INP	MATH.OUT
10 1 20 3 30 4 0 0	Case 1: 2 Case 2: 4 Case 3: 5

Hướng dẫn: Đối với bài toán này GV cần lưu ý cho HS trong việc đọc và ghi dữ liệu vì nó khác so với các bài toán khác. Dữ liệu vào gồm có nhiều test khác nhau và kết thúc file vào là một trường hợp n=0 và m=0. Với mỗi bộ test ở file vào thì lại phải có một kết quả ở file ra cho nên phần đọc và ghi dữ liệu ta phải làm như sau:

```
Begin
   assign(f,fi); reset(f);
   assign(g,fo); rewrite(g);
   so:=0;
   repeat
         read(f,n,m);
         if (m=0) and (n=0) then break;
         xu_ly;
   until false;

   close(f);
   close(g);
end.
```

Lưu ý cho HS nếu như dùng lệnh

```
repeat
         read(f,n,m);
         xu_ly;
until (m=0) and (n=0);
```

thì ở file ra sẽ như sau:

Case 1: 2

Case 2: 4
Case 3: 5
Case 4: 0
Bởi vì trong lệnh repeat – until điều kiện được kiểm tra sau khi thực hiện dãy các câu lệnh do đó trong file ra sẽ có thêm trường hợp case 4: 0. Chương trình của bài toán cụ thể như sau:

```
program maths;
uses crt;
const
    fi='C:\math.inp';
    fo='C:\math.out';
var
   n,m,so: integer;
   f,g: text;
Procedure xu_ly;
var a,b,dem: integer;
Begin
    dem:=0;
    for a:=1 to n-2 do
       for b:=a+1 to n-1 do
       if (a*a+b*b+m) mod (a*b)=0 then dem:=dem+1;
       so:=so+1;
    writeln(g,'case ', so, ': ', dem);
end;
Begin
   assign(f,fi); reset(f);
   assign(g,fo); rewrite(g);
   so:=0;
   repeat
         read(f,n,m);
         if (m=0) and (n=0) then break;
         xu_ly;
   until false;

   close(f);
   close(g);
end.
```

Bài 3: Stupid

File vào: stupid.in
File ra: stupid.out **Giới hạn thời gian: 1 giây**
File chương trình: stupid.pas **Giới hạn bộ nhớ: 64 KB**

Ở trường đại học *XYZ*, tất cả các sinh viên đều có mã các nhân là một số có 6 hoặc 7 chữ số. Nhưng nó không phải là một số bất kỳ. Chỉ các số có *tổng kiểm tra* với chữ số 0 ở hàng đơn vị thì có thể là mã các nhân hợp lệ.

Việc tính tổng kiểm tra của một mã cá nhân như sau: nhân lần lượt các chữ số của mã cá nhân từ phải sang trái với các số tương ứng 9, 7, 3. Sau đó cộng tất cả các tích lại với nhau. Ví dụ:

Mã cá nhân	:	1	3	9	0	2	7	2
Thừa số	:	9	7	3	9	7	3	9
Tích	:	9	21	27	0	14	21	18

Do đó tổng kiểm tra là 9 + 21 + 27 + 0 + 14 + 21 + 18 = 110, chữ số hàng đơn vị là 0, vì vậy mã này là hợp lệ. Đôi khi có những sinh viên viết chữ rất xấu, vì vậy giáo viên khó xác định được mã cá nhân. Bạn có nhiệm vụ giúp đỡ họ trong trường hợp đặc biệt này, ở đó có đúng 1 chữ số không đọc được. Trong trường hợp này, chữ số không đọc được là tính được (luôn có chính xác 1 chữ số đúng). Chú ý rằng lúc đầu các sinh viên viết rất tập trung, do đó chữ số đầu tiên là luôn đọc được và khác 0.

Dữ liệu: File vào gồm một dòng chứa một mã cá nhân với một chữ số được sửa bằng một dấu hỏi chấm và có độ dài 6 hoặc 7 chữ số.

Kết quả: File ra gồm một dòng ghi mã cá nhân đúng.

Ví dụ:

stupid.in	stupid.out
13?0272	1390272
3?5678	335678
345?78	345778
314?592	3146592

Hướng dẫn:

_ Đối với bài toán này vì trong dữ liệu vào có dấu '?' nên phải dùng dữ liệu kiểu xâu.

_ Đặc biệt quan trọng với bài toán này là việc chuyển từ kí tự sang số ta phải dùng hàm `ord(a[k])-48`

Chương trình cụ thể như sau:

```
program stupid;
uses crt;
const
fi='C:\stupid.inp';
fo='C:\stupid.out';
     thua_so:array[0..2] of integer = (9,7,3);
Var   a: string;
f: text;
Procedure doc;
Begin
assign(f,fi); reset (f);
readln(f,a);
close(f);
end;
```

Thủ tục xử lý có thể viết theo một trong 2 cách sau:

Cách 1:

```
Procedure xu_ly;
var i,k,n,j,sum: longint;
so: char;
Begin
n:=length(a);
for i:=1 to  n do  if a[i]='?' then
     Begin
j:=i;
for so:='0' to '9' do
       Begin
a[j]:=so;
sum:=0;
for k:=1 to n do
sum:=sum + (ord(a[k])-48)*thua_so[(k-1) mod 3];
if sum mod 10 =0 then break;
end;
end;
end;
```

Cách 2:

```
n:=length(a);
for i:=1 to  n do  if a[i]='?' then break;
```

```
for so:='0' to '9' do
     Begin
a[i]:=so;
sum:=0;
for j:=1 to n do
sum:=sum + (ord(a[j])-48)*thua_so[(j-1) mod 3];
if sum mod 10 =0 then break;
end;
end;
Procedure ghi;
Begin
assign(f,fo); rewrite(f);
writeln(f,a);
close(f);
end;
Begin
doc;
xu_ly;
ghi;
end.
```

Lưu ý: Trong mảng thua_so ta bắt buộc phải dùng là [0..2]

```
thua_so:array[0..2] of integer = (9,7,3);
```

mà không thể dùng [1..3] bởi vì khi tính tổng ta dùng đến số dư khi chia hết cho 3 (mà một số khi chia cho 3 chỉ có 3 số dư là 0, 1, 2): `thua_so[(j-1) mod 3];`

Bài 4: Số chính

File vào SOCHINH.INP

File ra SOCHINH.OUT

File chương trình SOCHINH.PAS

Giới hạn thời gian 1 giây

Cho dãy gồm n số nguyên số a_1, a_2, ..., a_n. Hãy tìm số xuất hiện nhiều lần nhất trong dãy.

Dữ liệu: Dòng đầu tiên của file vào chứa số nguyên dương n ($n \le 1.000.000$). Dòng thứ i trong số n dòng tiếp theo chứa một số nguyên a_i ($0 \le a_i \le 10.000$, $1 \le i \le n$).

Kết quả: File ra bao gồm một dòng duy nhất chứa số xuất hiện nhiều nhất và số lần xuất hiện của nó trong dãy. Nếu có nhiều số như vậy, hãy đưa ra số lớn nhất.

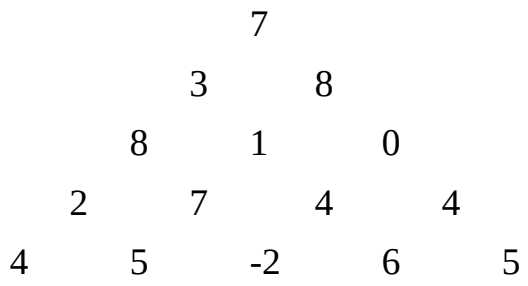
Ví dụ:

SOCHINH.INP	SOCHINH.OUT
6 5 1 8 2 8 5	8 2

```
Program So_chinh;
Uses crt;
Const
  fi = 'C:\sochinh.inp';
  fo = 'C:\sochinh.out';
Var
  dem : array[ 0..10000] of longint;
  f     : text;
Procedure doc;
Var i, n, so : longint;
Begin
  fillchar(dem, sizeof(dem), 0);
  assign(f, fi); reset(f);
  read(f, n);
  for i := 1 to n do
    begin
      read(f, so);
      dem[ so] := dem[ so ] + 1;
    end;
  close(f);
End;
Procedure ghi;
Var i, main : longint;
Begin
  main := 0;
  for i := 1 to 10000 do
```

```
   if dem[ i] >= dem[ main] then main := i;
  assign(f, fo); rewrite(f);
  writeln(f, main, ' ', dem[ main] );
  close(f);
End;
Begin
  doc;
  ghi;
End.
```

Bài 5: TAM GIÁC SỐ (Đề thi HSG Hà Tây năm 2006)



Hình trên mô tả một tam giác số có số hàng N=5. Đi từ đỉnh (số 7) đến đáy tam giác bằng một đường gấp khúc, mỗi bước chỉ được đi từ số ở hàng trên xuống một trong hai số đứng kề bên phải hay bên trái ở hàng dưới, và tính tích các số trên đường đi lại ta được một tích.

Ví dụ: Đường đi 7 8 1 4 6 có tích là S=1344, đường đi 7 3 1 7 5 có tích là S=735.

Yêu cầu: Cho tam giác số, tìm tích của đường đi có tích lớn nhất

Dữ liệu vào từ file văn bản TGS.INP:

- Dòng đầu tiên chứa số nguyên n (0<n<101)
- N dòng tiếp theo, từ dòng thứ hai đến dòng thứ N+!: Dòng thứ i có (i - 1) số cách nhau bởi dấu cách (Các số có giá trị tuyệt đối không vượt quá 100)

Kết quả: Đưa ra file văn bản TGS.OUT một số nguyên - là tích lớn nhất tìm được

TGS.INP	**TGS.OUT**
5 7 3 8 8 1 0 2 7 4 4 4 5 -2 6 5	**5880**

***TÀI LIỆU THAM KHẢO**

1, Tài liệu chuyên tin quyển 2 – Hồ Sỹ Đàm chủ biên
2, Giáo trình Giải thuật và lập trình – Tác giả Lê Minh Hoàng
3, Một số tài liệu khác của các đồng nghiệp