ĐẶNG TUẤN THÀNH
(Sư tầm và biên soạn)

21 GIỜ HỌC LẬP TRÌNH
Quyển 2

Tháng 01 năm 2021

LỜI NÓI ĐẦU

Nhằm cung cấp tài liệu dạy học môn Tin học trong trường phổ thông và phục vụ bồi dưỡng học sinh giỏi, tác giả sưu tầm và biên soạn cuốn học liệu: "**21 giờ học lập trình**" và "**Kỹ thuật lập trình**".

Tác giả luôn khắc ghi và biết ơn sự quan tâm của các đơn vị Phòng Giáo dục, trường THCS, trường THPT trong Tỉnh đối với việc áp dụng nguồn học liệu này đồng thời đã có những ý kiến góp ý quý báu, kịp thời để tác giả hoàn thiện học liệu.

Nguồn học liệu mới này được viết bởi ngôn ngữ lập trình Pascal và C++. Nội dung gồm **90** bài tập được biên soạn dạng bài kiểm tra, đề thi tương ứng với 21 giờ tự học. Hệ thống kiến thức được trình bày từ dễ đến khó, tạo điều kiện thuận lợi cho việc tự học, tự nghiên cứu. Với mỗi đơn vị kiến thức, có các bài tập, hướng dẫn thuật toán và code chương trình tham khảo bằng **Pascal** và **C++** giúp người học nhanh chóng làm quen định dạng, quy cách làm bài thi; Người học được hướng dẫn thuật toán có mức độ hoàn thiện bài khác nhau, phát triển kỹ năng làm mịn chương trình.

Trong thời gian tới, tác giả viết tiếp tài liệu mới ***"Ngôn ngữ lập trình Python"*** với các chuyên đề về ngôn ngữ lập trình Python và bài tập ứng dụng để thay thế dần ngôn ngữ Pascal. Rất mong các bạn đón đọc.

Tài liệu này được biên soạn rất tỉ mỉ nhưng không tránh được những thiếu sót, tác giả rất mong được sự đóng góp nội dung, phản hồi của bạn đọc vào email: dtthanh.c3ntt@yenbai.edu.vn.

Tác giả

MỤC LỤC

Ngày 1

1) Tìm max4.*

Nhập số nguyên a, b, c, d (a, b, c, d ≤ 10^9). Tìm giá trị lớn nhất của a, b, c, d.

Ý tưởng:

Xem số đầu tiên là số lớn nhất, mang giá trị lớn nhất này so sánh lần lượt với các giá trị còn lại, phần tử nào lớn hơn nó thì cập nhật lại nó.

```
#include <bits/stdc++.h>
using namespace std;
long int a, b, c, d, res;
int main()
{
   freopen("MAX4.inp","r",stdin);
   freopen("MAX4.out","w",stdout);
   cin >> a >> b >> c >> d;
   res = a;
   if(res < b)
      res = b;
   if(res < c)
      res = c;
   if(res < d)
      res = d;
   cout << res;
   return 0;
}
```

```
var a,b,c,d,max:longint;
begin
      assign(input,'max4.inp'); reset(input);
      assign(output,'max4.out');
rewrite(output);
      readln(a,b,c,d);
      max:= A;
      if b>max then max:=b;
      if c>max then max:=c;
      if d>max then max:=d;
      write(max);
      close(input); close(output);
end.
```

2) Phương trình bậc nhất EXP.*

Cho a, b nguyên (a, b ≤ 10^9). Hãy giải và biện luận phương trình ax + b = 0.

- Nếu phương trình vô nghiệm thông báo **None**.
- Nếu phương trình vô số nghiệm thông báo **Multiple**.
- Nếu phương trình có nghiệm, đưa ra dạng **x = -b/a**; **a, b nguyên tố cùng nhau**.

Ý tưởng:

Nếu a = 0 và b = 0 thì thông báo Multiple;

Nếu a = 0 và b!= 0 thì thông báo None;

Nếu a != 0 thì thực hiện:

+ lưu dấu của biểu thức vào biến **dau**.

+ a = |a|, b = |b|;

+ Tìm UCLN của a, b.

+ Kết quả bài toán là cout << (-1)*dau*b<< "/" << a;

```
#include <bits/stdc++.h>

using namespace std;
int a,b,m,ts,ms;

int main()
{
   freopen("EXP.inp","r",stdin);
   freopen("EXP.out","w",stdout);
   cin>>a>>b;

   if (a == 0 && b == 0) cout<<"Multiple";
   else if(a == 0 && b != 0) cout<<"None";
   if(a != 0)
   {
      int dau;
      if(a*b>0) dau=1;
      else dau=-1;
      a = abs(a);
      b = abs(b);
      int tam=__gcd(a,b);
      a = a / tam;
      b /= tam;
      cout<<-1*dau*b<<"/"<<a;
   }
   return 0;
}
```

```
var a,b,dau,t:longint;
function UCLN(m,n:longint):longint;
var r:longint;
begin
      while n<>0 do
            begin
                  r:=m mod n;
                  m:=n;
                  n:=r;
            end;
      exit(m);
end;
begin
      assign(input,'exp.inp'); reset(input);
      assign(output,'exp.out'); rewrite(output);
      readln(a,b);
      if a=0 then
            if b<>0 then write('None')
else write('Multiple')
    else
       begin
        if a*b>=0 then dau:=1
          else dau:= -1;
                  a:=abs(a);
                  b:=abs(b);
                  t:=UCLN(a,b);
a:=a div t;
b:=b div t;
write((-1)*dau*b,'/',a);
            end;
      close(input); close(output);
```

	end.

3) Tích các chữ số FNUM.*

Cho số nguyên dương N là số có 4 chữ số. Tìm tích các chữ số của N.

Ví dụ: N = 1234 → tich =24.

Ý tưởng:

Sử dụng phép toán chia nguyên và chia dư ta lần lượt tác các chữ số của N ra, lưu vào các biến a, b, c và d. Kết quả res = a * b * c * d;

```
#include <bits/stdc++.h>

using namespace std;
int i,s,n;
int main()
{
   freopen("FNUM.inp","r",stdin);
   freopen("FNUM.out","w",stdout);
   cin>>n;
   s = 1;
   while(n!=0)
   {
      i = n % 10;
      n = n / 10;
      s = s * i;
   }
   cout<<s;
   return 0;
}
```

```
var S,a,n:longint;
begin
      assign(input,'fnum.inp'); reset(input);
      assign(output,'fnum.out');
rewrite(output);
      readln(n);
      S:=1;
      while n<>0 do
            begin
                  a:=n mod 10;
                  n:=n div 10;
                  S:=S*a;
            end;
      write(S);
      close(input);
      close(output);
end.
```

```
#include <bits/stdc++.h>
using namespace std;
int a,b,c,d,N,res;
int main()
{
   freopen("FNUM.inp","r",stdin);
   freopen("FNUM.out","w",stdout);
   cin>>N;
```

```
    a=(N/1000);
    b=(N/100)%10;
    c=(N%100/10);
    d=(N%10);
    res=a*b*c*d;
    cout<<res;
    return 0;
}
```

4) Tìm X, Y FXY.*

Nhập số nguyên dương X, Y (X, Y ≤ 10^9).

Nếu X chẵn thì giảm X đi một nửa và tăng Y gấp đôi.

Ngược lại, thì tăng X một đơn vị và tăng Y thêm một lượng là X.

Đưa X, Y ra.

```
#include <bits/stdc++.h>

using namespace std;
int x,y;
int main()
{
    freopen("FXY.inp","r",stdin);
    freopen("FXY.out","w",stdout);
    cin>>x>>y;
    if(x%2==0)
        {
            x=x/2;
            y=y*2;
        }
    else
        {
            x+=1;
            y+=x;
        }
    cout<<x<<" "<<y;
```

```
var x,y:longint;
begin
    assign(input,'fxy.inp'); reset(input);
    assign(output,'fxy.out'); rewrite(output);
    readln(x,y);
    if x mod 2=0 then
        begin
            X:=X div 2;
            Y:=Y*2;
        end
        else
        begin
            X:=X+1;
            Y:=Y+X;
        end;
    write(x,#32,y);
    close(input);
    close(output);
end.
```

return 0; }	

5) Xóa số NDEL.*

Cho số nguyên dương N là số có 4 chữ số. Jame thực hiện xóa đi 1 số trong 4 chữ số của N thu được số N mới là số có 3 chữ số. Hãy tìm N mới lớn nhất có thể.

Ví dụ: N = 2123 → N = 223;

N = 3121 → N = 321.

Ý tưởng:

N = 1234, khi xoá 1 trong 4 số của N, ta có các số thu được là: 123, 124, **234**, 134

Như vậy với số N là số có 4 chữ số, ta tách được các số là res1, res2, res3, res4.

Bài toán quy về bài toán tìm giá trị lớn nhất của 4 số này.

```
#include <bits/stdc++.h>
#include <math.h>
using namespace std;
int n, d11, d22, d33, d44, a, b, c, d, res, i;
int main()
{
   freopen("NDEL.inp","r",stdin);
   freopen("NDEL.out","w",stdout);
   cin >> n;
   d11 = n / 1000;
   d22 = n / 100 % 10;
   d33 = n % 100 / 10;
   d44 = n % 10;
   a = d11 * 100 + d22 * 10 + d33;
   b = d11 * 100 + d33 * 10 + d44;
   c = d11 * 100 + d22 * 10 + d44;
   d = d22 * 100 + d33 * 10 + d44;
   res = max(max(a,b),max(c,d));
   cout << res;
   return 0;
}
```

```
uses math;
const fi='NDEL.inp';
     fo='NDEL.out';
var a,b,c,d,n:longint;
   rest,rest1,rest2,rest3,rest4:longint;
 begin
   assign(input,fi);reset(input);
   assign(output,fo);Rewrite(output);
   Readln(n);
   A:=N div 1000;
   B:=(N div 100) mod 10;
   C:=(N div 10) mod 10;
   D:=N mod 10;
   rest1:=100*b+10*c+d;
   rest2:=100*a+10*c+d;
   rest3:=100*a+10*b+d;
   rest4:=100*a+10*b+c;
rest := max(max(rest1,rest2),max(rest3,rest4));
   write(rest);
   close(input);
   close(output);
end.
```

6) Tìm ước GCD.*

Nhập số nguyên dương N . Liệt kê các ước nguyên tố của N.

Ví dụ: N = 10; Các ước nguyên tố 2, 5.

Ý tưởng:

- **Cách 1**: $N \leq 10^4$.

  Cho for (i,1,N), nếu N chia hết cho i và i là số nguyên tố thì đưa i ra.

```
#include <bits/stdc++.h>
using namespace std;
int nto(int u)
{
   if (u <= 1) return 0;
   if (u == 2 || u == 3) return 1;
   for (int i = 2;i <= trunc(sqrt(u)); i++)
      if (u % i == 0)
         return 0;
   return 1;
}
int n;
int main()
{
   freopen("GCD.inp","r",stdin);
   freopen("GCD.out","w",stdout);
   cin>>n;
   for (int i = 1; i <= n; i++)
      if (n % i == 0 && nto(i) == 1)
         cout<<i<<" ";
}
```

```
var n,i:longint;
function NT(x:longint):boolean;
var kt:boolean;
   i:longint;
begin
      kt:=true;
      if x<2 then kt:=false
        else if x>3 then
            for i:=2 to x div 2 do
                  if x mod i=0 then
                        begin
                              kt:=false;
                              break;
                        end;
      exit(kt);
end;
Begin
      assign(input,'GCD.inp'); reset(input);
      assign(output,'GCD.out'); rewrite(output);
      readln(n);
      for i:=1 to n do
            if NT(i) and (n mod i=0) then
write(i,#32);
      close(input);
      close(output);
end.
```

Cách 2: $N \leq 10^6$

Ta sàng nguyên tố Eratosthenes tới N, thu được mảng F[i] = 0/1 tương ứng với i không là số nguyên tố hoặc i là số nguyên tố.

Khi đó ta duyệt For(I,1, $\sqrt{N}$) nếu N chia hết cho I thì:

- Nếu I nguyên tố thì đẩy I vào mảng C.
- Nếu I != N / I và N/I nguyên tố thì đẩy N/I vào mảng C.

Sắp xếp mảng C thành dãy không giảm, đưa mảng C ra.

```
#include <bits/stdc++.h>
using namespace std;
int N,i,F[1000000];
void snt(int u)
{
   for (int i=1;i<=u;i++)
   F[i]=1;
   F[1]=0;
   for (int i=2;i<=u/i;i++)
      if (F[i]==1)
      for (int j=2;j<=u/i;j++)
      F[j*i]=0;
}
int main()
{
freopen("gcd.inp","r",stdin);
freopen("gcd.out","w",stdout);
cin>>N;
snt(N+1);
for (i=1;i<=N;i++)
   if (N%i==0 && F[i]==1)
cout<<i<<" ";
   return 0;
}
```

Cách 3: Phân tích N thành tích các thừa số nguyên tố.

Khi đó, các thừa số nguyên tố khác nhau trong cách phân tích này chính là các ước nguyên tố của N.

7) Tính tổng ước SDIV.*

Nhập N. Hãy tính tổng các ước nguyên dương của N.

Ví dụ: N = 12 → tong = 1 + 2 + 3 + 4 + 6 + 12 = 28.

Ý tưởng:

Cách 1: Duyệt For(I,1,n) nếu N chia hết cho I thì đẩy I vào tổng S.

Đưa tổng S ra.

```
#include <bits/stdc++.h>

using namespace std;
int n,s;
int main()
{
   freopen("SDIV.inp","r",stdin);
   freopen("SDIV.out","w",stdout);
   cin>>n;
   s = 0;
   for(int i=1;i<=n;i++)
      if(n % i == 0) s += i;
   cout<<s;
   return 0;
}
```

```
var i,n,s:longint;
begin
      assign(input,'sdiv.inp'); reset(input);
      assign(output,'sdiv.out'); rewrite(output);
      readln(n); s:=0;
      for i:=1 to n do
            if n mod i=0 then s:=s+i;
      write(s);
      close(input);
      close(output);
end.
```

Cách 2:

Duyệt For (I, 1, $\sqrt{N}$) nếu N chia hết cho I thì :

- Đẩy I vào tổng S.
- Nếu I != N/I thì đẩy N/I vào tổng S.

Đưa tổng S ra.

Lưu ý: S có thể lớn hơn N, nên chọn kiểu dữ liệu tốt hơn.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int tonguoc(int u)
{
   int tam = 0;
```

```
const
fi = 'sdiv.inp';
fo = 'sdiv.out';
var n,i,s : longint;
begin
      assign(input,fi); reset(input);
```

```
    for(int i = 1; i <= trunc(sqrt(u)); i++)
        if (u % i == 0)
    {
        tam = tam + i;
        if (u/i != i) tam = tam + u/i;
    }
    return tam;
}
int main()
{
    freopen("sdiv.inp","r",stdin);
    freopen("sdiv.out","w",stdout);
    cin >> n;
    cout << tonguoc(n);
    return 0;
}
```

```
        assign(output,fo); rewrite(output);
        readln(N); s:=0;
        for i:= 1 to trunc(sqrt(n)) do
        if n mod i=0 then
        begin
                s:= s + i;
                if i <> n div i then
                s:= s + n div i;
        end;
        write(s);
close(input); close(output);
end.
```

8) Phương trình bậc 2 EXP2.*

Nhập số nguyên a, b, c (a!=0). Giải và biện luận phương trình $ax^2 + bx + c = 0$

Ý tưởng:

Tính $d = b^2 - 4ac$;

Xét dấu của d. Nếu d < 0 thì thông báo 'None' ngược lại

{

Tính x1, x2;

Đưa x1, x2 ra.

}

```
#include <bits/stdc++.h>
using namespace std;
int a, b, c;
float x11,x22;
int main()
{
    freopen("exp2.inp","r",stdin);
    freopen("exp2.out","w",stdout);
```

```
const
        fi='EXP2.inp';
        fo='EXP2.out';
var a,b,c :longint;
    d,x1,x2: real;
begin
assign(input,fi);reset(input);
assign(output,fo);rewrite(output);
```

```
    cin >> a >> b >> c;
    int d = b*b -4*a*c;
    if (d < 0) cout << "None";
    else
    {
        x11 = (-b + sqrt(d))/(2*a);
        x22 = (-b - sqrt(d))/(2*a);
        printf("%0.2f %0.2f", x11, x22);
    }
    return 0;
}
```

```
readln(a,b,c);
        d:=sqr(b)-4*a*c;
        if (d<0) then
                write('None')
        else
                begin
                x1:=(-b+sqrt(d))/(2*a);
                x2:=(-b-sqrt(d))/(2*a);
                write(x1:0:2,#32,x2:0:2);
                end;
                close(input);close(output);
end.
```

Ngày 2

9) Số lớn nhất FMAX.*

Cho a, b, c, d nguyên dương có 1 chữ số. Hãy ghép các số thành số có 4 chữ số có giá trị lớn nhất. Ví dụ: a = 1; b = 2; c = 3; d = 4 → N = 4321.

Ý tưởng:

Để thu được số có giá trị lớn nhất thì các số a, b, c, d phải được sắp xếp thành dãy không tăng. Ta lưu a, b, c, d vào một mảng một chiều có 4 phần tử để tiện sắp xếp.

Ngoài ra ta có thể ghép thủ công các số này thành một số có 4 chữ số. Ta thu được 24 số (4!). Sau đó ta đi tìm max của dãy số này.

```
const
   fi='FMAX.inp';
   fo='FMAX.out';
var i,j,tam : longint;
      a:array[1..100000] of longint;
begin
      assign(input,fi);reset(input);
      assign(output,fo);rewrite(output);
      for i:= 1 to 4 do
            read(a[i]);
      for i:= 1 to 3 do
      for j:= i+1 to 4 do
      if a[i]<a[j] then
            begin
                  tam:=a[i];
                  a[i]:=a[j];
                  a[j]:=tam;
            end;
      for i:=1 to 4 do
write(a[i]);
      close(input);close(output);
end.
```

```
#include <bits/stdc++.h>
#define nmax 4
using namespace std;
int a[nmax];
int main()
{
   freopen("fmax.inp","r",stdin);
   freopen("fmax.out","w",stdout);
   for (int i=1;i<=4;i++)
      cin>>a[i];
   sort(a,a+4+1);
   for (int i=4;i>=1;)
      cout<<a[i];
   return 0;
}
```

10) Tổng chữ số SCOUNT.*

Cho số nguyên dương N là số có 5 chữ số. Hãy tính tổng các chữ số ở vị trí chẵn và tổng các chữ số ở vị trí lẻ.

Ví dụ: N = 1**6**3**1**5 → schan = 6 + 1 = 7; sle = 1 + 3 + 5 = 9

Ý tưởng: Đây là bài toán tác các chữ số của một số, ta lưu lần lượt vào các biến a, b, c, d, e. Khi đó tổng các số ở vị trí chẵn là b + d; tổng các số ở vị trí lẻ là: a + c + e.

```
#include <bits/stdc++.h>
#define nmax 5

using namespace std;
int a[nmax],sc,sl,j,h,n;
int main()
{
   freopen("SCOUNT.inp","r",stdin);
   freopen("SCOUNT.out","w",stdout);
   cin>>n;
   h = 1;
   while(n != 0)
   {
      j = n % 10;
      n /= 10;
      a[h++] = j;
   }
   sl = a[1]+a[3]+a[5];
   sc = a[2]+a[4];
   cout<< sc << " " <<sl;
   return 0;
}
```

```
var i,n,a,sochan,sole:longint;
begin
      assign(input,'scount.inp'); reset(input);
      assign(output,'scount.out');
rewrite(output);
      readln(n);
sochan:=0; sole:=0;
      while n>0 do
            begin
                  a:=n mod 10;
                  n:=n div 10;
                  i:=i+1;
                  if i mod 2=0 then
sochan:=sochan+a
                     else sole:=sole+a;
            end;
      write(sochan,#32,sole);
      close(input); close(output);
end.
```

11) Đổi tiền XCHAN.*

Cho X đồng. Các tờ tiền có mệnh giá 50, 20 và 10. Hãy đổi X đồng thành các tờ tiền sao cho số lượng tờ tiền thu được là ít nhất. Nếu không có phương án đổi thì ghi ra -1.

Ví dụ: X = 160 → soto = 4 (gồm 3 tờ 50 và 1 tờ 10)

X = 161 → soto = -1.

Ý tưởng:

Để thu được số lượng tờ tiền ít nhất, ta sẽ rút nhiều nhất có thể bằng mệnh giá 50, phần còn lại ta tiếp tục rút với mệnh giá 20. Sau đó, phần dư còn lại ra rút với mệnh giá 10. Sau khi rút bằng

các mệnh giá trên, nếu tiền vẫn còn dư thì chứng tỏ X đó không có phương án rút, ta ghi ra -1. Ngược lại ghi ra tổng số tờ tiền đã rút.

```
#include <bits/stdc++.h>
using namespace std;
int x, res;
int main()
{
   freopen("XCHAN.inp","r",stdin);
   freopen("XCHAN.out","w",stdout);
   cin >> x;
   int a = x / 50;
   x = x - 50 * a;
   int b = x / 20;
   x = x - 20 * b;
   int c = x / 10;
   x = x - 10 * c;
   if(x > 0)
      cout <<  -1;
   else
   {
      if (x == 0)
      {
         res = a + b + c;
         cout << res;
      }
   }
   return 0;
}
```

```
const
      fi='XCHAN.inp';
      fo='XCHAN.out';
      var X,to50,to20,to10 : longint;
begin
      assign(input,fi);reset(input);
      assign(output,fo);rewrite(output);
      readln(X);
      to50:=X div 50;
      X:=X-to50*50;
      to20:=X div 20;
      X:=X-to20*20;
      to10:=X div 10;
      X:=X-to10*10;
      if(X>0) then write(-1)
      else write(to50+to20+to10);
      close(input);close(output);
end.
```

12) Đổi tiền 2 XCHAN2.*

Cho X đồng. Các tờ tiền có mệnh giá 50, 20 và 10. Hãy đưa ra các phương án đổi X đồng thành các tờ tiền có mệnh giá như trên. Nếu không có phương án đổi thì ghi ra -1.

Ý tưởng:

Gọi a là số lượng tờ tiền mệnh giá 50, b là số lượng tờ tiền mệnh giá 20 và c là số lượng tờ tiền mệnh giá 10. Khi đó, để đổi X đồng ta có: 50.a + 20.b + 10.c = X. (1)

Giả sử ta đổi X đồng hết sang mệnh giá 50, thì số tờ 50 là a ≤ X/50. Tương tự thế, ta có b ≤ X/20 và c ≤ X/10. Từ đó ta duyệt qua toàn bộ giá trị của a, b và c, nếu phương trình (1) thoả mãn thì đưa ra a+b+c ra.

Nếu không có phương án đổi ta ghi ra -1 bằng kĩ thuật cầm cờ.

```
Const
      fi = 'XCHAN2.inp';
      fo = 'XCHAN2.out';
Var
n,i,l50k,l20k,l10k,max50,max20,ma
x10,du,dem,m:longint;
Begin
      assign(input,fi); reset(input);
      assign(output,fo); rewrite(output);
      Readln(n);
      max50:=n div 50;        max20:=n div 20;
max10:=n div 10;
      dem:=0;
      For l50k:=0 to max50 do
         For l20k:=0 to max20 do
            For l10k:=0 to max10 do
            If      (n=l50k*50+l20k*20+l10k*10)
then
            Begin
            Writeln(l50k,#32, l20k,#32 ,l10k);
            dem:= 1;
            m := 0;
            End;
      if dem = 0 then write(-1);
      close(input); close(output);
End.
```

```
#include <bits/stdc++.h>
using namespace std;
int n,t5,t2,t1;
int main()
{
freopen("XCHAN2.inp","r",stdin);
   freopen("XCHAN2.out","w",stdout);
   cin>>n;
   if(n % 10 != 0) cout<<-1;
   else
   {
      for(int t5=0;t5<=n/50;t5++)
         for(int t2=0;t2<=n/20;t2++)
            for(int t1=0;t1<=n/10;t1++)
               if(t5*50+t2*20+t1*10 == n)
                  cout<<t5<<"              "<<t2<<"
"<<t1<<endl;
   }
   return 0;
}
```

13) Đếm ước CDIV.*

Nhập số nguyên dương N và K. Liệt kê các số nhỏ hơn N có đúng K ước. Nếu không có số nào thì ghi ra -1.

Ví dụ: N = 10; K = 3; Kết quả: 4 9.

N = 10; K = 5; Kết quả: -1.

Ý tưởng:

Tổ chức chương trình có hàm đếm số lượng ước của u. Khi đó, duyệt các số thuộc đoạn [1, N-1], đến số thứ I, nếu souoc(i) = k thì đưa i ra.

Lưu ý xử lí khi không có số nào thoả mãn ghi ra -1.

```
const
      fi='CDIV.inp';
      fo='CDIV.out';
var N,i,K,p : longint;
      function demuoc(u:longint):longint;
      var i,dem: longint;
      begin
            dem:=0;
            for i:= 1 to trunc(sqrt(u)) do
            if u mod i=0 then
            begin
                  dem:=dem+1;
                  if u/i <> i then
                  dem:= dem+1;
            end;
            exit(dem);
      end;
begin
      assign(input,fi);reset(input);
      assign(output,fo);rewrite(output);
      readln(N,K);
      p:=0;
      for i:=1 to N-1 do
      if demuoc(i)=K then
      begin
            write(i,#32);
            p:=1;
      end;
     if p=0 then write(-1);
```

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,k,dem,a[nmax],s;
int uocn(int u)
{
   dem = 0;
   for(int j=1;j<=trunc(sqrt(u));j++)
      if(u % j == 0)
      {
         dem++;
         if(j != u / j) dem++;
      }
   return dem;
}
int main()
{
   freopen("CDIV.inp","r",stdin);
   freopen("CDIV.out","w",stdout);
   cin>>n>>k;
   s = 0;
   for(int i=1;i<=n-1;i++)
      if(uocn(i) == k)
      {
         cout<<i<<" ";
         s = 1;
      }
   if(s == 0) cout<<-1;
```

close(input);close(output); end.	return 0; }

14) Liệt kê số nguyên tố LPRIME.*

Cho a, b. Hãy liệt kê các số nguyên tố trong đoạn [a,b]. Nếu không có số nào thì ghi ra -1.

Ví dụ: a = 1; b = 8 → Kết quả: 2 3 5 7.

Ý tưởng:

Subtask1: Tổ chức hàm kiểm tra u có là số nguyên tố. Khi đó, duyệt các số nguyên trong đoạn [a,b], nếu I là số nguyên tố thì đưa I ra.

Độ phức tạp thuật toán: O($b\sqrt{b}$) chạy được với b ≤ 10^4.

Subtask2: Ta sàng nguyên tố tới b, thu được mảng f[i] =0/1 với I không là số nguyên tố/ I là số nguyên tố.

Khi đó, duyệt các số nguyên trong đoạn [a,b], nếu f[I]=1 thì đưa I ra.

Độ phức tạp thuật toán: O(b) chạy được với b ≤ 10^6.

```
const
    fi='LPPRIME.inp';
    fo='LPPRIME.out';
var a,b,p,i: longint;
function nguyento(u:longint):longint;
var i :longint;
begin
    if u<=1 then exit(0);
    if(u=2) or(u=3) then exit(1);
    for i:=2 to trunc(sqrt(u)) do
    if u mod i=0 then exit(0);
    exit(1);
    end;
begin
    assign(input,fi);reset(input);
    assign(output,fo);rewrite(output);
    readln(a,b);
    p:=0;
    for i:=a to b do
```

```
#include <bits/stdc++.h>
using namespace std;
bool nt(int u)
{
   if (u <= 1)
      return false;
   if (u == 3 || u == 2)
      return true;
   for (int i = 2;i<= trunc(sqrt(u));i++)
   if (u % i == 0)return false;
   return true;
}
int a,b,d;
int main()
{    freopen("LPPRIME.inp","r",stdin);
   freopen("LPPRIME.out","w",stdout);

   int d = 0;
   cin >> a >> b;
```

```
      if nguyento(i)=1 then
      begin
            write(i,#32);
            p:=1;
      end;
      if p=0 then write(-1);
      close(input);close(output);
end.
```

```
   for (int i = a; i <= b; i++) {
      if (nt(i) == 1)
      {
         cout << i << " ";
         d++;
      }
   }
   if (d == 0) {
      cout << -1;
   }
   return 0;
}
```

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int f[nmax],a,b,n;
void sangto(int u)
{
   for (int i=1;i<=u;i++) f[i] = 1;
   f[1] = 0;
   for(int i=2;i<=u/i;i++)
     if (f[i]==1)
       for(int j=2;j<=u/i;j++)
          f[i*j]=0;
}
int main()
{
   freopen("LPPRIME.inp","r",stdin);
   freopen("LPPRIME.out","w",stdout);
   cin>>a>>b;
   sangto(b);
   int dem = 0;
   for(int i=a;i<=b;i++)
      if(f[i] == 1) dem++;
```

```
    if(dem == 0) cout<<-1;
    else for(int i=a;i<=b;i++)
            if(f[i] == 1) cout<<i<<" ";
    return 0;
}
```

15) Ước chung UC3N.*

Cho số nguyên dương M, N và P. Hãy liệt kê các ước chung của M, N và P.

Ví dụ: M = 4, N = 6, P = 14 → Kết quả: 1 2

Ý tưởng: I là ước của M, N và P thì I thuộc đoạn [1, min(m,n,p)]. Ta for downto để tìm I đầu tiên thoả mãn và đưa I ra.

```
const
        fi = 'uc3n.inp';
        fo = 'uc3n.out';
var m,n,p,res,rmin,i : longint;
begin
        assign(input,fi); reset(input);
        assign(output,fo); rewrite(output);
        readln(m,n,p);
                rmin := m;
                if n < rmin then rmin := n;
                if p < rmin then rmin := p;
                for i := 1 to rmin do
                        if (m mod i = 0) and (n mod i =
0) and (p mod i = 0 ) then
                write(i,#32);
        close(input); close(output);
end.
```

```
#include <bits/stdc++.h>

using namespace std;
int m,n,p,a;
int main()
{
    freopen("UC3N.inp","r",stdin);
    freopen("UC3N.out","w",stdout);
    cin>>m>>n>>p;
    a = min(m,min(n,p));
    for(int i=1;i<=a;i++)
        if(m % i == 0 && n % i == 0 && p % i ==
0) cout<<i<<" ";
    return 0;
}
```

16) Tính P CALP.*

Tính P = a^x với x, a nguyên dương. Vì P có thể rất lớn, nên ta chỉ cần lưu kết quả chia dư cho 10^9+7.

Ý tưởng:

Subtask1: Cho I chạy từ 1 tới x ta tính p = p * a % L; // với L = 10^9+7.

Subtask2: Ta nhận thấy $P = a^{10} = a^5.a^5.a^0$; ta tính $a^5 = a^2.a^2.a^1$; $a^2 = a^1.a^1.a^0$. Ta có $a^1 = a$, $a^0 = 1$. Như vậy, để tính a^{10} ta tính a^5, a^2, a^1, a^0.

Ngoài ra, theo tính chất phép nhân đồng dư: p = (a.b)%c = ((a%c).(b%c))%c; //% = mod Do đó, sẽ tránh tràn kiểu dữ liệu khi tính số mũ lớn. Ví dụ 10^{100}.

Ta tổ chức chương trình con dạng hàm tính u^v.

```
const
    fi = 'calp.inp';
    fo = 'calp.out';
    L = 1000000007;
var x,a : longint;
    function amux (a1,x1 : longint) : longint;
    var tam : longint;
    begin
        if x1 = 0 then exit(1);
        if x1 = 1 then exit(a1);
        tam := amux(a1,x1 div 2);
        exit((tam mod L) * (tam mod L) * amux(a1,x1 mod 2)) mod L);
    end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(x,a);
        write(amux(a,x));
    close(input); close(output);

end.
```

```
#include <bits/stdc++.h>
#define l 1000000007

using namespace std;
int a,x;
long long int P;
long long int amux(int u, int v)
{
   long long int temp;
   if (v == 0)  return(1);
   if (v == 1)  return(u);
   temp = amux(u,v / 2);
   return((((temp*temp) % l)*amux(u,v % 2) % l) % l);

}
int main()
{
   freopen("CALP.inp","r",stdin);
   freopen("CALP.out","w",stdout);
   cin>>x>>a;
   P = amux(a,x);
   cout<<P;
   return 0;
}
```

Ngày 3

17) Số chính phương SQUARENU.*

Một số được gọi là số chính phương nếu $\sqrt{N} = [\sqrt{N}]$ (Căn bậc hai của N = phần nguyên của căn bậc 2 của N).

Cho số nguyên dương A và K. Liệt kê các số chính phương < A và chia hết cho K. Nếu không có số nào thì ghi ra -1.

Ý tưởng:

Subtask1: Duyệt qua tất cả các số thuộc đoạn [1, A-1]. Nếu I là số chính phương và I chia hết cho K thì đưa I ra.

Subtask2: Nhận xét, U chia hết cho K tức là U là một bội số của K và U < A. Vậy, ta đi tìm các bội số của K nhỏ hơn K bằng cách thử các số I thuộc đoạn [1, (A-1)/K]. Khi đó nếu i*K là số chính phương thì i*K là một nghiệm.

```
const
    fi = 'squarenu.inp';
    fo = 'squarenu.out';
var a,p,i,k : longint;
    function cp( u : longint) : boolean;
    begin
        exit(sqrt(u) = trunc(sqrt(u)));
    end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(a,k);
        p := 0;
        for i := 1 to a-1 do
            if (i mod k = 0) and (cp(i)) then
                begin
                    write(i,#32);
                    p := 1;
                end;
        if p = 0 then write(-1);
    close(input); close(output);
end.
```

```
#include <bits/stdc++.h>

using namespace std;
int n,k;
int dem=0;
int scp(int u)
{
   if(sqrt(u) == trunc(sqrt(u))) return 1;
   return 0;
}
int main()
{
  freopen("SQUARENU.inp","r",stdin);
freopen("SQUARENU.out","w",stdout);
   cin>>n>>k;
   for(int i=1;i<=n-1;i++)
      if (scp(i) == 1 && i % k == 0) dem++;
   if(dem == 0 ) cout<<-1;
  /* cách 2
for(int i=1;i<=(n-1)/k;i++)
        if (scp(i*k) == 1) cout<<i*k<<" ";
*/
```

	return 0; }

18) Cho số nguyên dương a. APRIME.*

Hãy đếm số lượng số nguyên tố N thỏa mãn a < N < 2a.

Subtask1: Tổ chức hàm kiểm tra u có là số nguyên tố. Khi đó, duyệt các số nguyên trong đoạn [a+1,2a-1], nếu I là số nguyên tố thì tăng res lên.

Độ phức tạp thuật toán: $O(2a\sqrt{2a})$ chạy được với a ≤ 10^4.

Subtask2: Ta sàng nguyên tố tới 2a, thu được mảng f[i] =0/1 với I không là số nguyên tố/ I là số nguyên tố.

Khi đó, duyệt các số nguyên trong đoạn [a-1,2a-1], nếu f[I]=1 thì đưa I ra.

Độ phức tạp thuật toán: $O(2a)$ chạy được với a ≤ 10^6.

```
const
    fi = 'aprime.inp';
    fo = 'aprime.out';
var i,a,res,N : longint;
    function nguyento( u : longint) : boolean;
    var i : longint;
    begin
        if u <= 1 then exit(False);
        if (u=2) or (u=3) then exit(true);
        for i := 2 to trunc(sqrt(u)) do
            if u mod i = 0 then exit(False);
            exit(True);
    end;
begin
    assign(input,fi); reset(input);
    assign(output,fo); rewrite(output);
    readln(N);
        for i := a+1 to 2*a-1 do
            if nguyento(i) then
            res := res + 1;
            write(res);
    close(input);close(output);
```

```
#include <bits/stdc++.h>

using namespace std;
int a,b,f[1000000];
void sangnt(int u)
{
   for (int i=1;i<=u;i++) f[i] = 1;
   f[1] = 0;
   for(int i=2;i<=u/i;i++)
     if (f[i]==1)
       for(int j=2;j<=u/i;j++)
          f[i*j]=0;
}
int main()
{
   freopen("APRIME.inp","r",stdin);
   freopen("APRIME.out","w",stdout);
   int dem=0;
   cin>>a;
   sangnt(2*a);
   for(int i=a+1;i<a*2;i++)
      if (f[i] == 1) dem++;
```

end.	cout<<dem; return 0; }

19) Số nguyên tố gần N NPRIME.*

Cho số nguyên dương N. Gọi A là số nguyên tố lớn nhất bé hơn N, B là số nguyên tố bé nhất lớn hơn N.

Cho số nguyên dương N (N ≥ 3), hãy tìm A và B.

Ví dụ: N = 6 → A = 5, B = 7.

Ý tưởng: Tổ chức chương trình có hàm nguyento(u) kiểm tra u có là số nguyên tố.

Ta tìm số bên trái N: L = N – 1; chừng nào L chưa là số nguyên tố thì giảm L đi. Khi đó, nếu L > 1 thì L là số nguyên tố.

Ta tìm số bên phải N: R = N + 1; chừng nào R chưa là số nguyên tố thì tăng R lên. Khi đó, R là số nguyên tố cần tìm.

```
#include <bits/stdc++.h>


using namespace std;
int n,a,b;
int ngto(int u)
{
   if (u<=1) return 0;
   if (u == 2 || u == 3) return 1;
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0 ) return 0;
   return 1;
}
int main()
{
   freopen("NPRIME.inp","r",stdin);
   freopen("NPRIME.out","w",stdout);
   cin>>n;
   a = n-1;
   while(ngto(a) != 1)
```

```
    a--;
  b = n+1;
  while(ngto(b) != 1)
    b++;
  cout<<a<<" "<<b;
  return 0;
}
```

20) Ước chung nguyên tố lớn nhất DIVPRIME.*

Cho số nguyên dương A, B và C. Hãy tìm ước chung nguyên tố lớn nhất của A, B và C. Nếu không có ghi ra -1.

Ý tưởng:

Subtask1: I là ước của A, B và C thì I thuộc đoạn [1, min(A,B,C)]. Ta for downto để tìm I nguyên tố đầu tiên thoả mãn và đưa I ra. Nếu không có số nào thì ghi ra -1.

Subtask2: Ta chuẩn bị sẵn mảng F[i]=0/1 lưu trạng thái số I là số nguyên tố bằng giải thuật sàng nguyên tố.

I là ước của A, B và C thì I thuộc đoạn [1, min(A,B,C)]. Ta for downto nếu f[I]=1 đầu tiên thoả mãn và đưa I ra và break. Nếu không có số nào thì ghi ra -1.

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int a,b,c,d[1000],m,p;
int f[nmax];
void sangto(int u)
{
  for(int i=1;i<=u;i++) f[i] = 1;
  f[1] = 0;
  for(int i=2;i<=u/i;i++)
    if(f[i] == 1)
      for(int j=i;j<=u/i;j++)
        f[i*j] = 0;
}
int main()
{
  freopen("DIVPRIME.inp","r",stdin);
```

```
    freopen("DIVPRIME.out","w",stdout);
    cin>>a>>b>>c;
    m = min(a,min(b,c));
    sangto(m);
    for(int i=m;i>=1;i--)
       if (f[i] == 1 && a % i ==0 && b % i == 0 && c % i == 0)
       {
         cout<<i;
         break;
         p=1;
       }
    if (p == 1) cout<<-1;
    return 0;
}
```

21) Cho chơi với dãy số GAMESEQ.*

Cho dãy số nguyên dương ghi các số từ 1 tới N trên bảng. Jone sẽ chọn trước các số chia hết cho A, sau đó Jame sẽ chọn các số trong các số còn lại và chia hết cho B. Tổng điểm của mỗi người là tổng các số đã được người đó chọn. Ai có tổng điểm lớn hơn thì người đó thắng, tất nhiên nếu điểm bằng nhau thì hòa.

Cho N, A và B (A!= B). Hãy ghi ra tên người thắng cuộc nếu có người thắng hoặc ghi ra "equalizer" nếu tổng điểm mỗi người có được bằng nhau.

Ý tưởng: Ta có dãy 1, 2, 3, 4,…, N-1, N. Khi đó, Jone chọn S1 là tổng các số chia hết cho A, Jame tổng các số chia hết cho B.

Ta xây dựng mảng đánh dấu F[i] = 0/1 với I không/có chia hết cho A:

```
For (I,1,N)
    if (N%i == 0)
    {
            f[i] = 1;
            resA = res A + I;
    }
     else f[i] = 0;
```

Sau đó, Jame sẽ chọn các phần tử I có f[i] == 0 và I % B == 0:

```
For (I,1,N) if (F[i]==0) resB = resB + I;
```

```
#include <bits/stdc++.h>
using namespace std;
int a,n,b,A,B,d[100000];
int main()
{
   freopen("GAMESEQ.inp","r",stdin);
   freopen("GAMESEQ.out","w",stdout);
   cin>>n>>a>>b;
   A = 0;
   B = 0;
   if( a == 1 && b != 1) cout<<"A";
   else if(a != 1 && b == 1) cout<<"B";
   else
   {
      for(int i=1;i<=n;i++)
         d[i] = i;
      for(int i=1;i<=n;i++)
         if(d[i] % a == 0)
         {
            A = A + d[i];
            d[i] = 1;
         }
      for(int i=1;i<=n;i++)
         if(d[i] % b == 0)
            B = B + d[i];
      if(A < B) cout<<"B"<<" "<<B;
      else if(A > B) cout<<"A"<<" "<<A;
      else cout<<"equalizer";
   }
   return 0;
}
```

22) Giải biện luận PTBN2.*

Nhập a, b, c, d và m. Giải và biện luận phương trình $\frac{ax+b}{cx+d} = m$. (c, d!=0)

- Nếu phương trình có nghiệm thì đưa ra dạng P/Q, với P, Q là số nguyên tố cùng nhau.
- Nếu phương trình vô nghiệm ghi ra NONE.
- Nếu phương trình vô số nghiệm ghi ra MULTIPLE.

Ý tưởng:

Ta có tập xác định x ≠ -d/c; biến đổi phương trình về dạng A.x + B = 0; và giải biện luận phương trình này.

Lưu ý: trường hợp phương trình có 1 nghiệm và nghiệm này trùng TXĐ thì kết luận NONE.

Với phương trình có nghiệm ta rút gọn phân số bằng cách tìm UCLN của tử số và mẫu số rồi rút gọn. Lưu ý thêm về dấu của biểu thức để tránh đưa ra dạng : -6/-5 hay 6/-5.

```
#include <bits/stdc++.h>
using namespace std;
int a,b,c,d,m,n,p,q;
int main()
{
   freopen("PTBN2.inp","r",stdin);
   freopen("PTBN2.out","w",stdout);
   cin>>a>>b>>c>>d>>m;
   if(a == 0 && b == 0 && c == 1 && d == 1 && m == 0) cout<<"MULTIPLE";
   else{
   p = d*m - b;
   q = a - c*m;
   n = __gcd(p,q);
   p = p / n;
   q = q / n;
   if(p > 0 && q > 0) cout<<p<<"/"<<q;
   else if(p < 0 && q > 0) cout<<p<<"/"<<q;
   else if(p > 0 && q < 0) cout<<-p<<"/"<<-q;
   else if(p < 0 && q < 0) cout<<-p<<"/"<<-q;
   else if(q == 0) cout<<"NONE";
   else cout<<"MULTIPLE";}
   return 0;
}
```

23) Tìm N TIMN.*

Nhập X. Tìm N lớn nhất để S = 1 + 2 + 3 + … + N ≤ X.

Ý tưởng: Đây là bài tập lặp while..do

```
#include <bits/stdc++.h>


using namespace std;
int i,x,s;
int main()
{
   freopen("TIMN.inp","r",stdin);
   freopen("TIMN.out","w",stdout);
   cin>>x;
   s = 0;
   i = 0;
   while (s<=x)
   {   i++;
      s += i;
   }
   cout << i;
   return 0;
}
```

24) Số Fibonacci FIBO1.*

Nhập N. Biết F[1] = 1; F[2] = 1; Tính F[N] = F[N-1] + F[N-2].

Ý tưởng:

Subtask1: Bài tập dùng mảng một chiều.

Subtask2: Dùng kĩ thuật xoay vòng 3 biến f, f1, f2.

```
f1 = 1; f2 = 1;
for(I,3,N)
{
        f = f1+f2;
        f1=f2;
        f2=f;
}
Cout << f;
```

Subtask3: Ứng dụng nhân ma trận để tính số Fibonacci thứ N.

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} * \begin{pmatrix} F_{i-1} \\ F_i \end{pmatrix} = \begin{pmatrix} F_i \\ F_{i+1} \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^T * \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = \begin{pmatrix} F_T \\ F_{T+1} \end{pmatrix}$$

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int f[nmax],n;
int main()
{
   freopen("FIBO1.inp","r",stdin);
   freopen("FIBO1.out","w",stdout);
   cin>>n;
   f[1] = 1;
   f[2] = 1;
   for(int i=3;i<=n;i++)
      f[i]=f[i-1] + f[i-2];
   cout<<f[n];
   return 0;
}
```

Ngày 4

25) Tam giác TRIANGLE.*

Nhập a, b, c. Hãy kiểm tra a, b, c có là 3 cạnh của tam giác. Nếu có, cho biết đó là tam giác gì (Vuông: 1; Thường: 0) ngược lại thông báo -1.

Ý tưởng: Bài tập rẽ nhánh

```
#include <bits/stdc++.h>


using namespace std;
int a,b,c;
int main()
{
   freopen("TRIANGLE.inp","r",stdin);
   freopen("TRIANGLE.out","w",stdout);
   cin>>a>>b>>c;
   if(b+c>a && a+c>b && a+b>c)
   {
      if(a*a == b*b+c*c || c*c == a*a+b*b || b*b == a*a+c*c) cout<<"1";
      else cout<<"0";
   }
   else cout<<"-1";
   return 0;
}
```

26) Tìm bội chung nhỏ nhất BCNN.*

"Bộ chung nhỏ nhất của số nguyên dương a và b được tính theo công thức: $bcnn = \frac{a \times b}{ucln(a,b)}$*."*

Nhập số nguyên dương a, b. Hãy tính BCNN của a và b.

Ý tưởng:

Subtask1: Tổ chức hàm tìm ước chung lớn nhất của a và b. Khi đó, áp dụng công thức ta tìm BCNN.

Subtask2: Phân tính a và b thành các thừa số nguyên tố. Khi đó, kết quả của bài toán là tích các thừa số nguyên tố với số mũ lớn nhất trong các phân tích a và b.

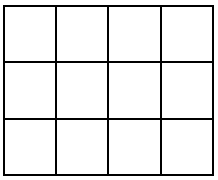
```
#include <bits/stdc++.h>
```

```
using namespace std;
int a,b;
int main()
{
   freopen("BCNN.inp","r",stdin);
   freopen("BCNN.out","w",stdout);
   cin>>a>>b;
   int p = __gcd(a,b);
   long long res = a*b / p;
   cout << res;
   return 0;
}
```

27) Hình chữ nhật HCN4.*

Trong hình chữ nhật được chia thành lưới ô vuông đơn vị. Cho biết x là số giao điểm bên trong hình chữ nhật, y là số cạnh của hình vuông đơn vị trong hình chữ nhật. Hãy tìm kích thước của hình chữ nhật.

Ý tưởng: Vẽ lưới hình chữ nhận ra. Gọi a, b là các cạnh của hình chữ nhật. Ta có:



X là số giao điểm bên trong hình chữ nhật: (b-1)(a-1) = X

Y là số cạnh bên trong hình chữ nhật: a(b-1) + (a-1)b = Y.

Rút a ở phương trình (1) thế vào phương trình (2) ta được phương trình bậc 2 một ẩn. Ta giải ra thu được b, từ đó tính được a. Đưa a, b ra.

28) Ước chung lớn nhất GCD3.*

Tìm ước chung lớn nhất của a, b và c nguyên dương.

29) Số Mersen MERSEN.*

Số P là số Mersen nếu P nguyên tố, P có thể phân tích thành $P = 2^r-1$, r là số nguyên tố.

Cho P, hãy kiểm tra P có là số Mersen hay không? Nếu có ghi ra 1 ngược lại ghi ra 0.

30) Xoá phần tử ARRDEL.*

Xóa các phần tử giống nhau đứng cạnh nhau trong dãy số, với những số này chỉ giữ lại đại diện một số. Đưa ra dãy sau khi xóa, giữ nguyên thứ tự đầu vào.

Ví dụ: N = 9; dãy 1 3 3 4 2 2 2 7 3 → Kết quả: 1 3 4 2 7 3

Dữ liệu: Vào từ tệp 'ARRDEL.INP' gồm:

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên $a_1, a_2, \ldots, a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'ARRDEL.OUT' dãy số còn lại sau khi xoá.

Mỗi số cách nhau một dấu cách.

ARRDEL.INP	**ARRDEL.OUT**
9 1 3 3 4 2 2 2 7 3	1 3 4 2 7 3
4 2 2 2 2	2

Ý tưởng: Thay vì việc xoá các phần tử giống nhau, ta đi "gắp" các phần tử khác nhau vào mảng C. Duyệt từ đầu đến cuối mảng, đến phần tử thứ I, nếu a[i] != a[i+1] thì đẩy a[i] vào mảng C.

Độ phức tạp: O(N)

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax],c[nmax];
int main()
{
   freopen("ARRDEL.inp","r",stdin);
   freopen("ARRDEL.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   int dem = 1;
   int h = 0;
   for(int i=1;i<=n;i++)
      if(a[i] == a[i+1]) dem++;
      else c[++h] = a[i];
   for(int i=1;i<=h;i++) cout<<c[i]<<" ";
   return 0;
}
```

31) Tìm K NFIND.*

Cho số nguyên dương M. Tìm số nguyên dương K nhỏ nhất sao cho tích các chữ số của K bằng M. Nếu không tồn tại K, đưa ra -1.

Dữ liệu: Vào từ tệp 'NFIND.INP' gồm số nguyên dương M ($M \leq 10^9$).

Kết quả: Ghi ra tệp 'NFIND.OUT' K nhỏ nhất tìm được hoặc ghi ra -1 nếu không tồn tại K.

NFIND.INP	NFIND.OUT
10	25
26	-1

Ý tưởng:

Tổ chức chương trình có hàm tính tích các chữ số của số u. Chừng nào tich(k) != M thì tăng K lên.

Cải tiến, ta phân tích M thành thừa số nguyên tố. Nếu có thừa số nguyên tố nào > 10 thì ghi ra -1. Ngược lại, ta ghép các thừa số nguyên tố thành một số nguyên. Đó là kết quả bài toán.

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int n,a[nmax];
long long res;
int main()
{
   freopen("NFIND.inp","r",stdin);
   freopen("NFIND.out","w",stdout);
   cin>>n;
   int i = 9;
   int k = 0;
   int p = 1;
   while(n > 1)
   {
      if(n % i == 0)
      {
         a[++k] = i;
         n /= i;
      }
      else i--;
      if(i <= 1) break;
   }
   for(int j=k;j>=1;j--) res = res*10 + a[j];
   if(n != 1) cout<<-1;
```

```
    else cout<<res;
    return 0;
}
```

32) Số Fibonacci nguyên tố FPRIME.*

Dãy số Fibonacci là dãy số có tính chất:

$F_1 = 1, F_2 = 1, F[N] = F[N-1] + F[N-2]$ $(\forall N \geq 3)$

Như vậy, các số Fibonacci đầu tiên là: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55…

Cho số nguyên dương P ($P \leq 10^9$). Hãy đếm số lượng các số Fibonacci nhỏ hơn hoặc bằng P và là số nguyên tố.

Ví dụ: P = 10 → Kết quả: 3

Thuật toán:

Tổ chức hàm nguyento(u). Lần lượt sinh các số Fibonacci, với mỗi số Fibonacci < P và là số nguyên tố ta tăng biến res lên 1 đơn vị. Đưa res ra.

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,f[nmax],a[nmax];
void sangto(int u)
{
    for(int i=1;i<=n;i++) f[i] = 1;
    f[1] = 0;
    for(int i=2;i<=u/i;i++)
        if(f[i] == 1)
            for(int j=i;j<=u/i;j++)
                f[i*j] = 0;
}
void fibo(int u)
{
    int dem = 0;
    a[1] = 1;
    a[2] = 1;
    for(int i=3;i<=u;i++)
```

```
        a[i] = a[i-1] + a[i-2];
    sangto(u);
    for(int i=1;i<=u;i++)
        if(f[a[i]] == 1) dem++;
    cout<<dem;
}
int main()
{
    freopen("FPRIME.inp","r",stdin);
    freopen("FPRIME.out","w",stdout);
    cin>>n;
    fibo(n);
    return 0;
}
```

Ngày 5

33) Sàng nguyên tố ERATOS.*

Cho số nguyên dương N ($N \leq 10^7$). Hãy liệt kê các số nguyên tố nhỏ hơn hoặc bằng N. Mỗi số cách nhau một dấu cách.

Ví dụ: N = 10 → Kết quả: 2 3 5 7.

Thuật toán:

Subtask1: For(I,1,N) nếu I nguyên tố thì đưa I ra.

Subtask2: Dùng sàng nguyên tố Eratosthenes để liệt kê các số nguyên tố nhỏ hơn hoặc bằng N.

```
#include <bits/stdc++.h>
#define nmax 10000005

using namespace std;
int n,f[nmax];
void sangto(int u)
{
   for(int i=1;i<=n;i++) f[i] = 1;
   f[1] = 0;
   for(int i=2;i<=u/i;i++)
      if(f[i] == 1)
         for(int j=i;j<=u/i;j++)
            f[i*j] = 0;
}
int main()
{
   freopen("ERATOS.inp","r",stdin);
   freopen("ERATOS.out","w",stdout);
   cin>>n;
   sangto(n);
   for(int i=1;i<=n;i++)
      if(f[i] == 1) cout<<i<<" ";
   return 0;
}
```

34) Vị trí đẹp ILUCKY.*

Cho N và dãy $a_1, a_2,\ldots,a_N$. Tìm các vị trí i (nếu có) chia dãy số thành 2 phần có tổng bằng nhau. Nếu không có thì ghi ra -1.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → i = 4 (Giải thích: Vị trí thứ 4 chia dãy thành hai phần có tổng 1+3+4+2=7+3).

Dữ liệu: Vào từ tệp 'ILUCKY.INP' gồm

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên $a_1, a_2,\ldots,a_N$ ($|a_i| \le 10^9$).

Kết quả: Ghi ra tệp 'ILUCKY.OUT' các vị trí đẹp tìm được, mỗi số cách nhau một dấu cách. Nếu không có vị trí đẹp thì ghi ra -1.

ILUCKY.INP	**ILUCKY.OUT**
6 1 3 4 2 7 3	4
4 1 2 3 7	-1

Ý tưởng:

Subtask1: For(I,1,N) với mỗi I ta tính tổng Strai (1,i) và Sphai(i+1,N). Nếu Strai = Sphai thì đưa I ra.

Subtask2: Gọi S[i] là tổng các phần tử liên tiếp từ 1 tới i.

Ta có, S[1]= a[1]; S[i] = S[i-1] + a[i]; //S[i] gọi là tổng tiền tố hoặc tổng tích luỹ.

Duyệt lại For (I,1,N), nếu S[i] = S[n]/2 thì đưa I ra.

Lưu ý: Do S[i] là tổng tích luỹ nên giá trị vượt kiểu dữ liệu của a[i].

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax];
long long s1,s2;
int main()
{
   freopen("ILUCKY.inp","r",stdin);
   freopen("ILUCKY.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++)
   {
```

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax];
long long s[nmax];
int main()
{

freopen("ILUCKY.inp","r",stdin);

freopen("ILUCKY.out","w",stdout);
   cin>>n;
```

```
    cin>>a[i];
    s2 += a[i];
  }
  int p = 0;
  for(int i=1;i<=n;i++)
  {
    s1 += a[i];
    s2 -= a[i];
    if(s1 == s2)
    {
      cout<<i;
      p = 1;
      break;
    }
  }
  if(p == 0) cout<<-1;
  return 0;
}
```

```
  for(int i=1;i<=n;i++)
  {
    cin>>a[i];
  }
  S[1] = a[1];
  for(int i=2;i<=n;i++)
    s[i]= s[i-1]+a[i];
int p = 0;
  for(int i=1;i<=n;i++)
      if (2*s[i] == S[n])
{
P = 1;
Cout << I << " ";
}
  if(p == 0) cout<<-1;
  return 0;
}
```

35) Giá trị nhỏ nhì PSECOND.*

Cho N và dãy a_1, a_2,…,a_N. Tìm giá trị nhỏ nhì (nếu có) của dãy số và đưa ra các vị trí đạt giá trị đó. Nếu không có thì ghi ra -1.

Ví dụ: N = 6; dãy 1 3 4 2 7 3 → min2 = 2; vitri = 4

Dữ liệu: Vào từ tệp 'PSECOND.INP' gồm

- Dòng 1: Ghi số nguyên dương N ($N \le 10^6$).
- Dòng 2: Ghi N số nguyên dương a_1, a_2,…, a_N ($a_i \le 10^9$).

Kết quả: Ghi ra tệp 'PSECOND.OUT' giá trị nhỏ nhì của dãy số và đưa ra các vị trí đạt giá trị đó. Mỗi số cách nhau một dấu cách. Nếu không có thì ghi ra -1.

PSECOND.INP	**PSECOND.OUT**
6 1 3 4 2 7 3	2 5
4 2 2 2 2	-1

Subtask1: $N \le 10^5$

Subtask2: $N \le 10^6$, $a_i \le 10^6$

Subtask3: $N \le 10^6$, $a_i \le 10^9$

Thuật toán:

Subtask1: Không mất tính tổng quát, ta sắp xếp dãy thành dãy không giảm (dãy tăng dần), lưu vị trị. Khi đó, a[1] là nhỏ nhất, a[N] là lớn nhất. Vậy, để tìm giá trị nhỏ nhì, ta duyệt từ 1 tới N, tìm phần tử đầu tiên khác a[1] đó là giá trị nhỏ nhì của dãy. Duyệt lại dãy, nếu a[i] = rmin2 thì đưa cs[i] ra.

Subtask2: Do $0 < a_i \le 10^6$, nên ta dùng phương pháp đếm phân phối a[i] vào mảng b[0…10^6]. Duyệt từ 1 tới N trên mảng b, nếu b[i] != 0 thứ 2 là giá trị nhỏ nhì, lưu vào rmin2. Duyệt từ 1 tới N trên mảng a, nếu a[i] = rmin2 thì đưa i ra.

Subtask3: Tìm giá trị nhỏ nhất của dãy lưu vào rmin1. Duyệt từ đầu đến cuối dãy, phần tử nào thoả mãn a[i] < rmin2 và a[i] != rmin thì rmin2 = a[i]. Đưa rmin2 ra. Duyệt từ 1 tới N trên mảng a, nếu a[i] = rmin2 thì đưa i ra.

Lưu ý xử lí trường hợp vô nghiệm bằng kĩ thuật cầm cờ.

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int n,a[nmax];
int main()
{
   freopen("PSECOND.inp","r",stdin);
   freopen("PSECOND.out","w",stdout);
   cin>>n;
   int res1 = 99999999;
   int res2 =-99999999;
   for(int i=1;i<=n;i++)
   {
      cin>>a[i];
      res1 = min(res1,a[i]);
      res2 = max(res2,a[i]);
   }
   if(res1 == res2) cout<<-1;
   else
   {
      for(int i=1;i<=n;i++)
         if(a[i] <= res2 && a[i] > res1) res2 = a[i];
```

```
    cout<<res2<<" ";
    for(int i=1;i<=n;i++)
      if(a[i] == res2) cout<<i<<" ";
  }
  return 0;
}
```

36) GCD2.*

Cho a, b nguyên dương có giá trị ≤ 10^9. Tìm ước chung lớn nhất của a và b.

Ví dụ: a = 4, b = 6 → KQ: 2.

Thuật toán: Áp dụng giải thuật Ơ'clit để tìm ước chung lớn nhất của a, b.

```
#include <bits/stdc++.h>
using namespace std;
int a,b;
int gcd(int u, int v)
{
        while(v > 0)
        {
            int r = u % v;
            u = v;
            v = r;
        }
        return u;
    }
    int main()
    {
        cin>>a>>b;
        int res = gcd(a,b);
        cout<<res;
        return 0;
}
```

37) TPS.*

Cho a, b, c, d nguyên có giá trị ≤ 10^9. Tính tổng và hiệu phân số a/b, c/d.

Kết quả ghi ra mỗi phân một dòng. Trong đó, mỗi dòng là phân số đã được rút gọn.

TPS.INP	TPS.OUT	Giải thích
2 3 6 6	5/3 -1/3	$\frac{2}{3}+\frac{6}{6}=\frac{10}{6}=\frac{5}{3}$; $\frac{2}{3}-\frac{6}{6}=\frac{-2}{6}=\frac{-1}{3}$;

Thuật toán:

Tổ chức chương trình thành chương trình con dạng hàm: UCLN.

Rút gọn phân số trước khi đưa kết quả ra.

Khi tính tổng và hiệu phân số lưu ý dấu của biểu thức phân số để tránh đưa ra dạng: -3/-5; 3/-5 mà phải đưa ra là: 3/5; -3/5.

```
#include <bits/stdc++.h>
using namespace std;
int a,b,c,d;
int daut,dauh;
int gcd(int u, int v)
{
   while(v > 0)
   {
      int r = u % v;
      u = v;
      v = r;
   }
   return u;
}
int main()
{
   cin>>a>>b>>c>>d;
   long long tst = a*d + b*c;
   long long ms = b * d;
   int tsh = a*d - b*c;
   int m = gcd(abs(tst),abs(ms));
   int n = gcd(abs(tsh),abs(ms));
   int rtt = tst / m;
   int rmt = ms / m;
   int rth = tsh / n;
```

```
    int rmh = ms / n;
    if(rtt * rmt >= 0) daut = 1;
    else int daut = -1;
    if(rth * rmh >= 0) dauh = 1;
    else dauh = -1;
    cout<<daut*abs(rtt)<<"/"<<abs(rmt)<<endl<<dauh*abs(rth)<<"/"<<abs(rmh);
    return 0;
}
```

38) Tongcs.*

Cho N nguyên dương $\leq 10^9$. Hãy tính tổng các chữ số của N.

Ví dụ: N = 1224; Kết quả: 9

Thuật toán:

Tổ chức hàm tính tổng các chữ số của N để sử dụng vào các bài toán khác.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int tcs(int u)
{
   int res = 0;
   while(u != 0)
   {
      res = res + (u % 10);
      u /= 10;
   }
   return res;
}
int main()
{
   cin>>n;
   cout<<tcs(n);
   return 0;
}
```

39) NPRDEL.*

Cho N nguyên dương ≤ 10^9. Xoá lần lượt từ phải sang trái. Tìm số nguyên tố lớn nhất có được. Biết phải xoá ít nhất 1 số. Nếu không có số nào ghi ra -1.

Ví dụ: N = 172 → Kết quả: 17.

N = 12 → Kết quả: -1.

Thuật toán:

Tổ chức hàm kiểm tra tính nguyên tố.

Lần lượt cắt chữ số ở hàng đơn vị, ta kiểm tra tính nguyên tố của N mới, nếu nó là số nguyên tố thì dừng lại và đưa nó ra.

Sử dụng biến P để đánh dấu có nghiệm/không có nghiệm: Kĩ thuật cầm cờ.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int ngto(int u)
{
   if(u <= 1) return 0;
   if(u == 2 || u == 3) return 1;
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0) return 0;
   return 1;
}
int main()
{
   cin>>n;
   int p = 1;
   while(ngto(n) == 0)
   {
      n /= 10;
      if(n == 0)
      {
         p = 0;
         break;
      }
      if(ngto(n) == 1) break;
```

```
        }
        if(p == 0) cout<<-1;
        else cout<<n;
        return 0;
}
```

40) NPALIN.*

Cho N nguyên dương ≤ 10^9 . Kiểm tra N có là số đối xứng. Nếu có ghi ra 1 ngược lại ghi ra 0.

Số đối xứng là số có số đảo bằng chính nó. Các số có một chữ số là số đối xứng.

Ví dụ: Số 1221 → Kết quả: 1

Thuật toán:

Tìm số đảo của N. Nếu số đảo của N bằng N thì đưa N ra.

Nên tổ chức thành hàm số đảo.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int sodao(int u)
{
    int res = 0;
    while(u != 0)
    {
        res = res * 10 +(u % 10);
        u /= 10;
    }
    return res;
}
int palin(int u)
{
    if(u != sodao(u)) return 0;
    return 1;
}
int main()
{
```

```
    cin>>n;
    if(palin(n)) cout<<1;
    else cout<<0;
    return 0;
}
```

41) SUBPALIN.*

Cho $N \leq 10^9$. Xoá phần lượt từ phải sang trái 1 kí tự. Hãy tìm số đối xứng lớn nhất có được. Biết nhất phải phải xoá ít nhất một chữ số.

Ví dụ: N = 1233211 → Kết quả: 123321

Thuật toán:

Tổ chức hàm kiểm tra tính đối xứng.

Lần lượt cắt chữ số ở hàng đơn vị, ta kiểm tra tính đối xứng của N mới, nếu nó là số nguyên tố thì dừng lại và đưa nó ra.

Sử dụng biến P để đánh dấu có nghiệm/không có nghiệm: Kĩ thuật cầm cờ.

```
#include <bits/stdc++.h>
using namespace std;
int n;
int sodao(int u)
{
    int res = 0;
    while(u != 0)
    {
        res = res * 10 +(u % 10);
        u /= 10;
    }
    return res;
}
int palin(int u)
{
    if(u != sodao(u)) return 0;
    return 1;
}
```

```
int main()
{
   cin>>n;
   int p = 1;
   while(palin(n) == 0)
   {
      n /= 10;
      if(palin(n) != 0) break;
   }
   if(p == 0) cout<<-1;
   else cout<<n;
   return 0;
}
```

42) NSINH.*

Cho N nguyên dương ≤ 10^9. Tìm số sinh của số N. Biết số sinh của N bằng tổng các chữ số của N nhân với chữ số lớn nhất của N.

Ví dụ: N = 1234 → Kq = (1+2+3+4)*(4) = 40

Thuật toán:

Tổ chức hàm tính tổng chữ số của N và hàm tìm chữ số lớn nhất của N.

```
#include <bits/stdc++.h>
using namespace std;
int n,kq;
int xuli(int u)
{
   int res = 0;
   int rmax = -10;
   while(u != 0)
   {
      int j = u % 10;
      res = res + j;
      u /= 10;
      rmax = max(rmax, j);
   }
```

```
        return res*rmax;
    }
    int main()
    {
        cin>>n;
        kq = xuli(n);
        cout<<kq;
        return 0;
}
```

Ngày 6

43) LCM.*

Cho a, b, c có giá trị ≤ 10^9. Tìm BCNN của a, b, c.

Ví dụ: a = 6, b = 8, c = 12 → Kết quả: 24

Thuật toán:

Phân tích a, b, c thành tích thừa số nguyên tố.

a = 6 = $2^1.3^1$	B = 2.2.2=2^3	C = 12 = 2.2.3=$2^2.3^1$
2 1	2 3	2 2
3 1		3 1

Kết quả: $2^{maxsomu2}.3^{maxsomu3} = 2^{3.}3^1 = 24$.

44) TSNT.*

Cho N nguyên dương ≤ 10^6. Phân tích N thành tích các thừa số nguyên tố.

Ví dụ: N = 70 = $2^1.5^1.7^1$ ghi ra dạng:

2 1

5 1

7 1

Thuật toán:

Sử dụng mảng C lưu nghiệm.

Chừng nào N chia hết cho i thì tăng biến đếm lên và giảm N = N/I;

Đẩy đếm vào C[i].

Nếu phần còn lại của N là số nguyên tố thì đẩy nốt phần còn lại vào mảng C.

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int n,j,c[nmax],rmax;
void ptts(int u)
{
   int i = 2;
   for(int i=2;i<=trunc(sqrt(u));i++)
   {
      int d = 0;
      while(u % i == 0)
         {
            d++;
```

```
                u /= i;
            }
        c[i] = d;
    }
    if(u>1)
    {
        c[u] = 1;
        rmax = max(rmax,u);
    }
}
int main()
{
    cin>>n;
    rmax = 0;
    ptts(n);
    if(rmax == 0) rmax = n/2;
    for(int i=2;i<=rmax;i++)
        if(c[i] != 0)cout<<i<<" "<<c[i]<<endl;
    return 0;
}
```

45) LCMLIST.*

Cho N nguyên dương và dãy số nguyên dương a_1, a_2, …, a_N. Tìm bội chung nhỏ nhất của N số nguyên trên. Kết quả ghi ra dạng các thừa số nguyên tố của BCNN.

Ví dụ: N = 3 và dãy 2, 3, 4 → KQ: 12 ($2.2.3 = 2^2.3^1$)

Ghi ra:

2 2

3 1

Thuật toán:

Với mỗi số a[i] ta phân tích a[i] thành thừa số nguyên tố, tuy nhiên thay đổi chút khi lưu c[i] = max(c[i],dem);

Tổ chức hàm tính amun(a,b) để tính a^b.

Duyệt trên mảng C, nếu c[i]!= 0 thì ta tính res = res * amub(i,c[i]);

```

```

46) Array1

[RMAXMIN.*] Cho N và dãy a_1, a_2, …, a_n. Đưa ra màn hình số nhỏ nhất và số lớn nhất trong mảng a[i].

Thuật toán:

Khởi tạo rmax, rmin.

Ta đi tìm rmax, rmin rồi đưa ra.

Ví dụ:

RMAXMIN.INP	RMAXMIN.OUT
5 2 4 6 8 9	2 9

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax],rmin,rmax;
int main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   rmin = 99999999;
   rmax = -99999999;
   for(int i=1;i<=n;i++)
      rmin = min(rmin,a[i]);
   for(int i=1;i<=n;i++)
      rmax = max(rmax,a[i]);
   cout<<rmin<<" "<<rmax;
   return 0;
}
```

47) Array2

[PMAXMIN.*] Cho N và dãy a_1, a_2, …, a_n. Đưa ra màn hình vị trí của số lớn nhất và vị trí của số nhỏ nhất trong mảng.

Ví dụ:

PMAXMIN.INP	PMAXMIN.OUT
4 2 5 1 2	2 3

Thuật toán:

Tìm rmax, rmin

Duyệt từ đầu đến cuối mảng, nếu a[i] = rmax thì đưa I ra.

Duyệt từ đầu đến cuối mảng, nếu a[i] = rmin thì đưa I ra.

```
#include <bits/stdc++.h>
#define nmax 1000007

using namespace std;
int n,a[nmax],rmin,rmax;
int main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   rmin = 99999999;
   rmax = -99999999;
   for(int i=1;i<=n;i++)
      rmin = min(rmin,a[i]);
   for(int i=1;i<=n;i++)
      rmax = max(rmax,a[i]);
   for(int i=1;i<=n;i++)
      if(rmin == a[i]) cout<<i<<" ";
   cout<<endl;
   for(int i=1;i<=n;i++)
      if(rmax == a[i]) cout<<i<<" ";
   return 0;
}
```

48) Array3

[CMAXMIN.*] Cho N và dãy a_1, a_2, …, a_n. Hãy đếm số phần tử nhỏ nhất và số phần tử lớn nhất trong mảng.

Ví dụ:

CMAXMIN.INP	CMAXMIN.OUT
5 2 4 6 8 9	1 1

Thuật toán:

Tìm rmax, rmin

Duyệt từ đầu đến cuối mảng, nếu a[i] = rmax thì tăng dmax lên

Duyệt từ đầu đến cuối mảng, nếu a[i] = rmin thì tăng dmin lên.

Đưa dmax và dmin ra.

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax];
int main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   int rmin = 99999999;
   int rmax = -99999999;
   for(int i=1;i<=n;i++)
      rmin = min(rmin,a[i]);
   for(int i=1;i<=n;i++)
      rmax = max(rmax,a[i]);
   int dmax = 0;
   int dmin = 0;
   for(int i=1;i<=n;i++)
      if(rmin == a[i]) dmin++;
   for(int i=1;i<=n;i++)
      if(rmax == a[i]) dmax++;
   cout<<dmin<<" "<<dmax;
   return 0;
```

```
}
```

49) Array4

[DMAXMIN.*] Cho N và dãy a_1, a_2, …, a_n. Hãy loại bỏ số lớn nhất và số nhỏ nhất trong mảng rồi cộng tổng các số còn lại.

Ví dụ:

DMAXMIN.INP	DMAXMIN.OUT
5 1 2 5 3 9	10

Thuật toán:

Tìm rmax, rmin.

Đếm số lượng rmax, rmin ta thu được dmax, dmin.

Tính tổng các phần tử trong mảng.

Kết quả là: res = S – dmax.rmax – dmin.rmin;

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax];
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    int rmin = a[1];
    int rmax = a[1];
    for(int i=1;i<=n;i++)
        rmin = min(rmin,a[i]);
    for(int i=1;i<=n;i++)
        rmax = max(rmax,a[i]);
    for(int i=1;i<=n;i++)
        if(rmin == a[i] || rmax == a[i]) a[i] = 0;
    long long s = 0;
```

```
    for(int i=1;i<=n;i++)
        s += a[i];
    cout<<s;
    return 0;
}
```

50) Array 5

[CK.*] Cho N, K và dãy a_1, a_2, …, a_n. Đếm số lượng các phần tử chia hết cho k.

Ví dụ:

CK.INP	CK.OUT
4 2 5 10 15 7	1

Thuật toán:

Duyệt từ đầu dãy đến cuối dãy, nếu a[i] chia hết cho K thì tăng đếm lên.

Đưa đếm ra.

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax],k;
int main()
{
    cin>> n >> k;
    for(int i=1;i<=n;i++) cin>>a[i];
    int dem = 0;
    for(int i=1;i<=n;i++)
        if(a[i] % k == 0) dem++;
    cout<<dem;
    return 0;
}
```

51) Array 6

[LCP.*] Cho N và dãy a_1, a_2, …, a_n. Hãy liệt kê các số chính phương có trong mảng. Nếu trong mảng không có số chính phương nào thì đưa ra màn hình -1.

Ví dụ:

LCP.INP	LCP.OUT
4 2 4 7 9	4 9
3 2 5 8	-1

Thuật toán:

Tổ chức hàm kiểm tra một số có là số chính phương.

Duyệt từ đầu dãy đến cuối dãy, nếu a[i] là số chính phương thì đưa a[i] ra.

Sử dụng kĩ thuật cầm cờ để kiểm soát vô nghiệm.

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax],k;
int main()
{
   cin>> n;
   for(int i=1;i<=n;i++) cin>>a[i];
   int p = 0;
   for(int i=1;i<=n;i++)
      if(sqrt(a[i]) == trunc(sqrt(a[i])))
      {
         cout<<a[i]<<" ";
         p = 1;
      }
   if(p == 0) cout<<-1;
   return 0;
}
```

52) Array7

[LLPRIME.*] Cho N và dãy a_1, a_2, …, a_n. Hãy liệt kê các số nguyên tố trong mảng ra, kết quả đã được sắp xếp từ bé đến lớn. Nếu trong mảng không có phần tử nào là số nguyên tố thì đưa ra màn hình -1.

Ví dụ:

LLPRIME.INP	LLPRIME.OUT
5 2 7 1 9 3	2 7 9
1 4	-1

Thuật toán:

Subtask1:

Tổ chức hàm kiểm tra tính nguyên tố

Duyệt từ đầu mảng đến cuối mảng, nếu a[i] là số nguyên tố thì đưa a[i] ra.

Độ phức tạp thuật toán: O(N.$\sqrt{amax}$), làm cách này học sinh đạt 60% điểm của bài.

Subtask2:

Tổ chức thủ tục Sàng nguyên tố thu được mảng F[i] lưu trạng thái nguyên tố của i. Ta sàng tới Rmax.

Khi đó, Duyệt từ đầu mảng đến cuối mảng, nếu F[a[i]] là số nguyên tố thì đưa a[i] ra.

Độ phức tạp thuật toán O(a_{max} + N).

```
#include <bits/stdc++.h>
#define nmax 1000007

using namespace std;
int n,a[nmax],f[nmax],c[nmax];
void sangto(int u)
{
   for(int i=1;i<=u;i++) f[i] = 1;
   f[1] = 0;
   for(int i=2;i<=u/2;i++)
      if(f[i] == 1)
         for(int j=i;j<=u/2;j++)
            f[i*j] = 0;
}
int main()
{
   int h = 0;
```

```
    cin>>n;
    int rmax = -999999999;
    for(int i=1;i<=n;i++)
    {
       cin>>a[i];
       rmax = max(rmax,a[i]);
    }
    sangto(rmax);
    for(int i=1;i<=n;i++)
       if(f[a[i]] == 1)
          c[++h] = a[i];
    sort(c+1,c+h+1);
    for(int i=1;i<=h;i++)
       cout<<c[i]<<" ";
    if(h == 0) cout<<-1;
    return 0;
}
```

Ngày 7

53) Array8

[CPPRIME.*] Cho N và dãy $a_1, a_2, \ldots, a_n$. Hãy đếm số cặp (i, j) sao cho $a_i + a_j$ là số nguyên tố (i!=j).

Ví dụ

CPPRIME.OUT	CPPRIME.OUT
4 1 2 3 4	4

Thuật toán:

Subtask1:

Tổ chức hàm kiểm tra tính nguyên tố

For(I,1,N-1)

For(j,i+1,N)

Nếu (a[i]+a[j]) là số nguyên tố thì tăng đếm lên.

Độ phức tạp thuật toán: $O(N^2.\sqrt{amax})$, làm cách này học sinh đạt 60% điểm của bài.

Subtask2:

Tổ chức thủ tục Sàng nguyên tố thu được mảng F[i] lưu trạng thái nguyên tố của i. Ta sàng tới 2.Rmax.

Khi đó, Duyệt từ đầu mảng đến cuối mảng, nếu F[a[i]] là số nguyên tố thì đưa a[i] ra.

Độ phức tạp thuật toán $O(a_{max} + N^2)$.

```
#include <bits/stdc++.h>
#define nmax 1000007

using namespace std;
int n,a[nmax],f[nmax],dem;
void sangto(int u)
{
   for(int i=1;i<=u;i++) f[i] = 1;
   f[1] = 0;
   for(int i=2;i<=u/2;i++)
      if(f[i] == 1)
         for(int j=i;j<=u/2;j++)
            f[i*j] = 0;
}
void sub1()
{
   dem = 0;
   int rmax=-9999999;
   for(int i=1;i<=n;i++)
      rmax = max(rmax,a[i]);
   sangto(2*rmax);
   for(int i=1;i<=n-1;i++)
      for(int j=i+1;i<=n;j++)
         if(f[a[i]+a[j]] == 1) dem++;
}
int main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   sub1();
   cout<<dem;
   return 0;
```

}

54) ArrayAdd1

Cho N, và dãy a_1, a_2, a_3,.., a_N

Liệt kê các dãy con có tổng lớn nhất

Đưa ra tổng lớn nhất

55) ArrayAdd2

Cho N, và dãy a_1, a_2, a_3,.., a_N

Liệt kê các dãy con có tổng số nguyên tố.

56) ArrayAdd3

Cho N, và dãy a_1, a_2, a_3,.., a_N

Chọn 3 phần tử có tổng là số nguyên tố lớn nhất

57) ArrayAdd4

Cho N, K và dãy a_1, a_2, a_3,.., a_N

Chọn k phần tử có tổng là số đối xứng lớn nhất.

58) ArrayAdd5

Cho N, và dãy a_1, a_2, a_3,.., a_N

Chọn ra một số phần tử là số Fibonaci.

59) ArrayAdd5

Cho N, và dãy a_1, a_2, a_3,.., a_N

Chọn một số phần tử có tổng lớn nhất sao cho không được chọn 3 phần tử liên tiếp.

60) PPS1

N21.*: Cho n. Liệt kê dãy nhị phân độ dài n không có 2 số 1 cạnh nhau

VD:

Input	Output
3	0 0 0 0 0 1 0 1 0 1 0 0 1 0 1

Thuật toán:

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1. Nếu điền đủ N ô thì đưa dãy vừa điền ra, rồi sinh cấu hình kế tiếp.

Độ phức tạp thuật toán: $O(2^N)$

```
#include <bits/stdc++.h>
#define nmax 20
using namespace std;

int n,a[nmax];
void xuli()
{
   int p = 0;
   for(int i=1;i<=n;i++)
      if(a[i] == 1 && a[i+1] == 1)
      {
         p = 1;
         break;
      }
   if(p == 0)
   {
      for(int i=1;i<=n;i++) cout<<a[i]<<" ";
      cout<<endl;
   }
}
void thu(int i)
{
   for(int j=0;j<=1;j++)
   {
      a[i] = j;
      if(i == n) xuli();
      else thu(i+1);
   }
}
int main()
{
```

```
    freopen("N21.inp","r",stdin);
    freopen("N21.out","w",stdout);
    cin>>n;
    thu(1);
    return 0;
}
```

61) PPS2

N22.*: Cho n và dãy a_1, a_2,…, a_n. Liệt kê các dãy con của dãy a.

Thuật toán:

Ta sinh dãy nhị phân có độ dài bằng N, song song với mảng a.

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1 của mảng b. Nếu điền đủ N ô thì duyệt từ 1 tới N, nếu b[i] = 1 thì chọn a[i] và đưa a[i] ra, sau đó sinh cấu hình kế tiếp.

Input	Output:
4 2 1 4 5	 5 4 4 5 1 1 5 1 4 1 4 5 2 2 5 2 4 2 4 5 2 1 2 1 5 2 1 4 2 1 4 5

```
#include <bits/stdc++.h>
#define nmax 20
```

```
using namespace std;

int n,a[nmax],b[nmax];
void xuli()
{
      for(int i=1;i<=n;i++)
         if(b[i] == 1) cout<<a[i]<<" ";
      cout<<endl;
}
void thu(int i)
{
   for(int j=0;j<=1;j++)
   {
      b[i] = j;
      if(i == n) xuli();
      else thu(i+1);
   }
}
int main()
{
   freopen("N22.inp","r",stdin);
   freopen("N22.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   thu(1);
   return 0;
}
```

62) PPS3

N23.*: Cho N và dãy $a_1,a_2,a_3,\ldots,a_N$. Liệt kê tổng của các dãy con của dãy a.

Thuật toán:

Ta sinh dãy nhị phân có độ dài bằng N, song song với mảng a.

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1 của mảng b. Nếu điền đủ N ô thì duyệt từ 1 tới N, nếu b[i] = 1 thì chọn a[i] và đẩy vào tổng S. sau đó đưa S ra và khởi tạo lại S = 0, sau đó sinh cấu hình kế tiếp.

Input	Output
3 3 1 2	0 2 1 3 3 5 4 6

```
#include <bits/stdc++.h>
#define nmax 20
using namespace std;

int n,a[nmax],b[nmax];
void xuli()
{
   int c[nmax];
   int s = 0;
   int h = 0;
   for(int i=1;i<=n;i++)
      if(b[i] == 1)
      {
         c[++h] = a[i]; // dùng mảng C[h] lưu các phần tử được chọn.
      }
   for(int i=1;i<=h;i++) s += c[i];
   cout<<s<<endl;
}
void thu(int i)
{
   for(int j=0;j<=1;j++)
   {
      b[i] = j;
      if(i == n) xuli();
```

```
        else thu(i+1);
    }
}
int main()
{
    freopen("N23.inp","r",stdin);
    freopen("N23.out","w",stdout);
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    thu(1);
    return 0;
}
```

63) PPS4

N24.*: Cho n, k và dãy a_1, a_2, …, a_n. Chọn 1 số phần từ sao cho tổng chia hết k.

Thuật toán:

Ta sinh dãy nhị phân có độ dài bằng N, song song với mảng a.

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1 của mảng b. Nếu điền đủ N ô thì duyệt từ 1 tới N, nếu b[i] = 1 thì chọn a[i] và đẩy vào tổng S, nếu S chia hết cho K thì duyệt từ 1 tới N, nếu b[i]=1 thì đưa a[i] ra. Khởi tạo lại S = 0, sau đó sinh cấu hình kế tiếp.

Input	Output
3 3 1 2	0 2 4 6

```
#include <bits/stdc++.h>
#define nmax 20
using namespace std;

int n,a[nmax],b[nmax],k,x;
void xuli()
{
```

```
    int c[nmax];
    long long s = 0;
    int h = 0;
    for(int i=1;i<=n;i++)
        if(b[i] == 1)
        {
            c[++h] = a[i];
        }
    for(int i=1;i<=h;i++) s += c[i];
    if(s % k == 0) cout<<s<<endl;
}
void thu(int i)
{
    for(int j=0;j<=1;j++)
    {
        b[i] = j;
        if(i == n) xuli();
        else thu(i+1);
    }
}
int main()
{
    freopen("N24.inp","r",stdin);
    freopen("N24.out","w",stdout);
    cin>>n>>k;
    for(int i=1;i<=n;i++) cin>>a[i];
    thu(1);
    return 0;
}
```

64) PPS5

N25.*: Cho n và dãy a_1, a_2 ,…, a_n. Chọn một số phần từ sao cho tổng là số nguyên tố lớn nhất. Đưa tổng tìm được ra. Nếu không có cách chọn nào thì ghi ra -1.

Input	Output

3 3 1 2	5

Thuật toán:

Ta sinh dãy nhị phân có độ dài bằng N, song song với mảng a.

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1 của mảng b. Nếu điền đủ N ô thì duyệt từ 1 tới N, nếu b[i] = 1 thì chọn a[i] và đẩy vào tổng S, nếu S là số nguyên tố thì cập nhật resmax = max(resmax,S) và khởi tạo lại S = 0, sau đó sinh cấu hình kế tiếp.

```
#include <bits/stdc++.h>
#define nmax 20
using namespace std;
int n,a[nmax],b[nmax],k,x;
long long res = -99999999;
int ngto(int u)
{
   if(u == 1) return 0;
   if(u == 2 || u == 3) return 1;
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0) return 0;
   return 1;
}
void xuli()
{
   int c[nmax];
   long long s = 0;
   int h = 0;
   for(int i=1;i<=n;i++)
      if(b[i] == 1)
      {
         c[++h] = a[i];
      }
   for(int i=1;i<=h;i++) s += c[i];
   if(ngto(s) == 1)
      res = max(res,s);
}
```

```
void thu(int i)
{
   for(int j=0;j<=1;j++)
   {
      b[i] = j;
      if(i == n) xuli();
      else thu(i+1);
   }
}
int main()
{
   freopen("N25.inp","r",stdin);
   freopen("N25.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   thu(1);
   cout<<res;
   return 0;
}
```

65) PPS6

N26.*: Cho n và dãy a_1, a_2, a_3,…, a_n. Chọn một số phần tử có tích lớn nhất.

Input	Output
3 3 1 2	6

Thuật toán:

Ta sinh dãy nhị phân có độ dài bằng N, song song với mảng a.

Cài đặt thuật toán đệ quy Thu(i); để thử điền vào ô thứ I giá trị 0 hoặc 1 của mảng b. Nếu điền đủ N ô thì duyệt từ 1 tới N, nếu b[i] = 1 thì chọn a[i] và đẩy vào tổng S và cập nhật resmax = max(resmax,S) và khởi tạo lại S = 0, sau đó sinh cấu hình kế tiếp.

Ngày 8

66) ARRTSNT

7, N27.*: Phân tích N thành thừa số nguyên tố.

Thuật toán:

Bài toán này ta có cài đặt theo cách khác đó là dùng mảng C[k] lưu số mũ, D[k] lưu thừa số nguyên tố.

Input	Output
28	2 2 7 1

```
#include <bits/stdc++.h>
#define nmax 1000000
using namespace std;
int n,d[nmax],c[nmax];
int k = 0;
void phantich(int u)
{
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0)
      {
         d[++k] = i;
         int dem = 0;
         while(u % i == 0 && u > 1)
         {
            dem++;
            u /= i;
         }
         c[k] = dem;
      }
      if(u > 1)
      {
         d[++k] = u;
         c[k] = 1;
      }
```

```
}
int main()
{
    cin>>n;
    phantich(n);
    for(int i=1;i<=k;i++) cout<<d[i]<<" "<<c[i]<<endl;
    return 0;
}
```

67) PPSTN

N29.*: Cho n và dãy $a_1,a_2,a_3,\ldots,a_n$. Liệt kê các dãy con tăng ngặt (là dãy con tăng không có phần tử bằng nhau).

Input	Output
5 1 4 4 2 3	1 4 1 4 1 2 3 2 3

Thuật toán:

Lần lượt sinh ra các cấu hình lưu vào mảng b[i]. Với mỗi cấu hình, nếu b[i] =1 thì đẩy a[i] vào mảng c[k]. Kiểm tra mảng C[k] có tăng ngặt không, nếu có thì đưa mảng C ra và khởi tạo lại K.

68) PPSLSEQ

N30LIST.*: Cho n và dãy $a_1,a_2,a_3,\ldots,a_n$. Tìm dãy con tăng dài nhất. Đưa ra độ dài.

Thuật toán:

Lần lượt sinh ra các cấu hình lưu vào mảng b[i]. Với mỗi cấu hình, nếu b[i] =1 thì đẩy a[i] vào mảng c[k]. Kiểm tra mảng C[k] có tăng ngặt không, nếu có thì cập nhật rmax = max(rmax,K) và khởi tạo lại K.

Input	Output
5 1 4 4 2 3	3

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax],c[nmax];
```

```
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    int dem = 1;
    int k = 0;
    for(int i=1;i<=n;i++)
        if(a[i] <= a[i+1]) dem++;
        else
        {
            c[++k] = dem;
            dem = 1;
        }
    int res = -99999999;
    for(int i=1;i<=k;i++) res = max(res,c[i]);
    cout<<res;
    return 0;
}
```

69) QHD1

QHD1.*: Cho N, P và dãy $a_1,a_2,..,a_n$

Cho P bộ số u và v. Hãy tính tổng các phần tử từ u tới v. Đưa tổng này ra.

QHD1.inp	QHD1.out
5 3 -1 2 -3 4 2 1 2 1 3 4 5	1 -2 6

Thuật toán:

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v ta đưa tong(u,v) ra.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta đưa X ra.

Độ phức tạp thuật toán: O(N+P).

```
#include <bits/stdc++.h>
#define nmax 100000
using namespace std;
int n,u,v,a[nmax],p;
void sub2()
{
   long long s[nmax];
   s[1] = a[1];
   for(int i=1;i<=n;i++)
      s[i] = s[i-1] + a[i];
   int u11,v11;
   for(int i=1;i<=p;i++)
   {
      cin>>u11>>v11;
      long long q = s[v11] - s[u11-1];
      cout<<q<<endl;
   }
}
int main()
{
   freopen("QHD1.inp","r",stdin);
   freopen("QHD1.out","w",stdout);
   cin>>n>>p;
   for(int i=1;i<=n;i++) cin>>a[i];
   sub2();
   return 0;
}
```

2, QHD2.*: Cho P cặp u,v. Tính S_{uv}. Liệt kê các đoạn [u,v] có tổng là số nguyên tố lớn nhất.

QHD2.inp	QHD2.out
5 4 10 2 5 3 12 1 2	17

1 3 4 5 2 4	

Thuật toán:

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số nguyên tố thì đưa tổng này ra.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta đưa X ra.

Độ phức tạp thuật toán: O(N+P).

```
#include <bits/stdc++.h>
#define nmax 100000

using namespace std;
int n,a[nmax],u,v,p;
int ngto(int u)
{
   if(u<=1) return 0;
   if(u == 2 || u == 3) return 1;
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0) return 0;
   return 1;
}
int main()
{
   freopen("QHD2.inp","r",stdin);
   freopen("QHD2.out","w",stdout);
   cin>>n>>p;
   for(int i=1;i<=n;i++) cin>>a[i];
   int s[nmax];
   s[0] = 0;
   for(int i=1;i<=n;i++)
```

```
        s[i] = s[i-1] + a[i];
    for(int i=1;i<=p;i++)
    {
        cin>>u>>v;
        long long q = s[v] - s[u-1];
        if(ngto(q)) cout<<q<<endl;
    }
    return 0;
}
```

70) QHD3

QHD3.*: Cho P cặp u,v. Tính S_{uv}. Tìm đoạn [u,v] có tổng là số nguyên tố lớn nhất.

QHD3.inp	QHD3.out
5 4 10 2 5 3 12 1 2 2 3 4 5 2 4	17

Thuật toán:

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số nguyên tố thì cập nhật Rmax.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta cập nhật Rmax.

71) QHD4

QHD4.*: Cho P cặp u,v. Tính S_{uv}. Tìm đoạn [u,v] có tổng có đúng K ước.

QHD4.inp	QHD4.out
5 4 4 10 2 5 3 12 1 2 2 3	15

4 5 2 4	

Thuật toán:

Tổ chức hàm đếm số lượng ước của số nguyên U.

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là có đúng K ước thì đưa tổng này ra.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X có đúng K ước thì đưa X ra.

72) QHD5

QHD5.*: Cho P cặp u,v. Tính S_{uv}. Tìm đoạn [u,v] có tổng là số đối xứng.

QHD5.inp	QHD5.out
5 4 4 10 2 5 3 12 1 2 2 3 4 5 2 4	7

Thuật toán:

Tổ chức hàm kiểm tra tính đối xứng của số nguyên U.

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số đối xứng thì đưa tổng này ra.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X là số đối xứng thì đưa X ra.

73) QHD6

QHD6.*: Cho P cặp u,v. Tính S_{uv}. Tìm đoạn [u,v] có tổng là số chính phương.

QHD4.inp	QHD4.out
5 4 10 2 5 3 13	16

1 2 2 3 4 5 2 4	

Thuật toán:

Tổ chức hàm kiểm tra tính chính phương của số nguyên U.

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số chính phương thì đưa tổng này ra.

Độ phức tạp thuật toán: O(P.N).

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X là số chính phương thì đưa X ra.

74) QHD7

QHD7.*: Cho N và dãy a_1, a_2,…, a_n. Tìm đoạn con liên tiếp có tổng là số chính phương. Đưa số chính phương đó.

QHD7.inp	QHD7.out
5 1 2 6 3 4	1 9 16 9 4

Thuật toán:

Tổ chức hàm kiểm tra tính chính phương của số nguyên U.

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số chính phương thì đưa tổng này ra.

Độ phức tạp thuật toán: $O(N^3)$.

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới I, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X là số chính phương thì đưa X ra.

Độ phức tạp thuật toán $O(N^2)$.

```
#include <bits/stdc++.h>
#define nmax 100000

using namespace std;
int n,a[nmax];
```

```
long long kq;
int scp(long long u)
{
   if(sqrt(u) == trunc(sqrt(u))) return 1;
   else return 0;
}
int main()
{
   freopen("QHD7.inp","r",stdin);
   freopen("QHD7.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   long long s[nmax];
   s[0] = 0;
   for(int i=1;i<=n;i++)
      s[i] = s[i-1] + a[i];
   for(int i=1;i<=n;i++)
      for(int j=i;j<=n;j++)
      {
         kq = s[j] - s[i-1];
         if(scp(kq) == 1) cout<<kq<<" ";
      }
   return 0;
}
```

75) QHD8

QHD8.*: Cho N và dãy $a_1,a_2,\ldots,a_n$. Tìm đoạn con liên tiếp có tổng là số nguyên tố lớn nhất. Đưa tổng đó.

QHD8.inp	QHD8.out
5 2 3 2 1 5	13

Thuật toán:

Tổ chức hàm kiểm tra tính nguyên tố của số nguyên U.

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) là số nguyên tố thì cập nhật Rmax.

Độ phức tạp thuật toán: $O(N^3.\sqrt{amax})$.

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới i, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X là số nguyên tố thì cập nhật Rmax.

Độ phức tạp thuật toán $O(N^2\sqrt{amax})$.

```
#include <bits/stdc++.h>
#define nmax 100000

using namespace std;
int n,a[nmax];
int ngto(long long u)
{
   if(u <= 1) return 0;
   if(u == 2 || u == 3) return 1;
   for(int i=2;i<=trunc(sqrt(u));i++)
      if(u % i == 0) return 0;
   return 1;
}
int main()
{
   freopen("QHD8.inp","r",stdin);
   freopen("QHD8.out","w",stdout);
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   long long s[nmax];
   s[0] = 0;
   for(int i=1;i<=n;i++)
      s[i] = s[i-1] + a[i];
   long long res = -trunc(1e18);
   for(int i=1;i<=n;i++)
      for(int j=i;j<=n;j++)
      {
         long long p = s[j] - s[i-1];
         if(ngto(p) == 1) res = max(res,p);
```

```
    }
  cout<<res;
  return 0;
}
```

76) QHD9

QHD9.*: Cho N và dãy $a_1,a_2,\ldots,a_n$. Tìm đoạn con liên tiếp có tổng chia hết cho K và dài nhất (l = v-u +1). Đưa ra độ dài đó.

QHD9.inp	QHD9.out
5 2 2 5 3 1 4	3

Thuật toán:

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v nếu tong(u,v) chia hết cho K thì cập nhật Lmax (v-u+1).

Độ phức tạp thuật toán: $O(N^3)$.

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới i, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu X chia hết cho K thì cập nhật Lmax (v-u+1).

Độ phức tạp thuật toán $O(N^2)$.

Ngoài ra, dự theo tính chất cộng đồng dư (a+b)%c = (a%c + b%c)%c.

Nên ta không nhất thiết phải lưu a[i] ngay từ lúc nhập dữ liệu, mà lưu a[i]%K.

Khi đó, tổng đoạn u, v không tràn kiểu dữ liệu.

```
#include <bits/stdc++.h>
#define nmax 100000

using namespace std;
int n,a[nmax],k,x;
int main()
{
  freopen("QHD9.inp,"r",stdin);
  freopen("QHD9.out,"w",stdout);
   cin>>n>>k;
   for(int i=1;i<=n;i++)
```

```
    {
        cin>>x;
        a[i] = x % k;
    }
    int s[nmax];
    s[0] = 0;
    for(int i=1;i<=n;i++)
        s[i] = s[i-1] + a[i];
    int res = -trunc(1e18);
    for(int i=1;i<=n;i++)
        for(int j=i;j<=n;j++)
        {
            int p = s[j] - s[i-1];
            if(p % k == 0) res = max(res,j - i + 1);
        }
    cout<<res;
    return 0;//
}
```

77) QHD10

QHD10.*: Cho N và dãy a_1, a_2,…, a_n. Tìm đoạn con liên tiếp dài nhất có tổng là số Fibonacci. Đưa ra độ dài đó.

QHD10.inp	QHD10.out
6 1 2 4 1 5 4	5

Thuật toán:

Chuẩn bị mảng Fibonacci tối đa khoảng 70 phần tử đầu tiên (mảng F[i] tăng).

Tổ chức hàm tìm kiếm nhị phân (x) có trong mảng F[i].

Subtask1: Tổ chức chương trình con dạng hàm tính tổng các phần tử từ u đến v của mảng a. Ở chương trình chính, với mỗi bộ u, v; x = tong(u,v); nếu TKNP(x) > 0 thì cập nhật Lmax (v-u+1).

Độ phức tạp thuật toán: $O(N^3.\log 70)$.

Subtask2: Gọi S[i] là tổng các phần tử từ 1 tới i, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v: X = S[v] – S[u-1];

Trong chương trình chính, mỗi bộ u, v ta tính X, nếu TKNP(x) > 0 thì cập nhật Lmax (v-u+1).

Độ phức tạp thuật toán $O(N^2.\log 70)$.

Ngoài ra, dự theo tính chất cộng đồng dư (a+b)%c = (a%c + b%c)%c.

Nên ta không nhất thiết phải lưu a[i] ngay từ lúc nhập dữ liệu, mà lưu a[i]%K.

Khi đó, tổng đoạn u, v không tràn kiểu dữ liệu.

```
#include <bits/stdc++.h>
#define nmax 100000

using namespace std;
int n,a[nmax];
long long f[nmax];
void fibo()
{
   f[1] = 1;
   f[2] = 1;
   for(int i=3;i<=50;i++)
      f[i] = f[i-1] + f[i-2];
}
int tknp(long long u)
{
   int dau = 1;
   int cuoi = 70;
   while(dau <= cuoi)
   {
      int giua = (dau+cuoi)/2;
      if(f[giua] == u) return giua;
      if(f[giua] > u) cuoi = giua - 1;
      else dau = giua + 1;
   }
   if(dau > cuoi) return 0;
}
int main()
{
   freopen("QHD10.inp,"r",stdin);
   freopen("QHD10.out,"w",stdout);
```

```
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    long long s[nmax];
    s[0] = 0;
    fibo();
    for(int i=1;i<=n;i++)
        s[i] = s[i-1] + a[i];
    long long res = -trunc(1e18);
    for(int i=1;i<=n;i++)
        for(int j=i;j<=n;j++)
        {
            long long p = s[j] - s[i-1];
            long long q = j - i + 1;
            if(tknp(p) > 0) res = max(res,q);
        }
    cout<<res;

    return 0;
}
```

11, QHD11.*: Cho N và dãy $a_1,a_2,\ldots,a_n$. Tìm vị trí i, j chia dãy thành 3 phần có tổng bằng nhau. Đưa i,j ra. Nếu không có thì ghi ra -1.

QHD11.inp	QHD11.out
5 1 4 5 4 1	2 3

Thuật toán:

Gọi S[i] là tổng các phần tử từ 1 tới i, ta có s[0] = 0; ta có S[i] = S[i-1] + a[i]; Lưu ý mảng S[i] kiểu long long (int64).

Khi đó, tổng đoạn từ u tới v:

S1 = S[i]-S[0]; S2=S[j]-S[i]; S3= S[n] – S[j];

Nếu S1=S2=S3 thì đưa I, j ra.

Độ phức tạp thuật toán $O(N^2)$.

```
#include <bits/stdc++.h>
#define nmax 100000
```

```
using namespace std;
int n,a[nmax];
int main()
{
   cin>>n;
   for(int i=1;i<=n;i++) cin>>a[i];
   long long s[nmax];
   s[0] = 0;
   for(int i=1;i<=n;i++)
      s[i] = s[i-1] + a[i];
   for(int i=1;i<=n;i++)
      for(int j=i+1;j<=n;j++)
      {
         long long s1 = s[i] - s[0];
         long long s2 = s[j] - s[i];
         long long s3 = s[n] - s[j];
         if(s1 == s2 && s2 == s3) cout<<i<<" "<<j;
      }
   return 0;
}
```

78) QHD12

QHD12.*: Cho N và dãy $a_1,a_2,\ldots,a_n$. Gọi R_i là số phần tử bên phải $\leq a_i$

Gọi L_i là số phần tử bên trái $\geq a_i$

Điểm của a_i là $| R_i - L_i |$. Tìm điểm lớn nhất của mỗi phần tử.

QHD12.inp	QHD12.out
5 3 1 2 4 5	4

Thuật toán:

Duyệt từ 1 tới N, với mỗi I ta đếm Ri và Li, và cập nhật rmax=max(rmax,abs(ri-li));

```
#include <bits/stdc++.h>
#define nmax 1000000

using namespace std;
int n,a[nmax];
```

```
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    int res = -trunc(1e9);
    for(int i=1;i<=n;i++)
    {
        int r1 = 0;
        for(int j=1;j<i;j++)
            if(a[i] >= a[j]) r1++;
        int r2 = 0;
        for(int j=i+1;j<=n;j++)
            if(a[j] >= a[i]) r2++;
        res = max(res,abs(r1-r2));
    }
    cout<<res;
    return 0;
}
```

Ngày 9

79) Xau1

Sxau1.*: Cho một xâu S.

- ✓ Đưa xâu S ra.
- ✓ Đưa độ dài xâu S.
- ✓ Sắp xếp các kí tự trong xâu S.

Sxau1.inp	Sxau2.out
adecb	adecb 5 abcde

```
#include <bits/stdc++.h>

using namespace std;
string s;
char temp;
int main()
{
   cin>>s;
   cout<<s<<endl;
   int m = s.length();
   cout<<s.length()<<endl;
   for(int i=1;i<=m-2;i++)
      for(int j=i+1;j<=m-1;j++)
         if(s[i] > s[j])
         {
            temp = s[i];
            s[i] = s[j];
            s[j] = temp;
         }
   for(int i=0;i<=m-1;i++) cout<<s[i];
   return 0;
}
```

80) Xau2

Sxau2.*: Cho xâu S. Đếm kí tự ‘a’ trong xâu. Xóa các kí tự ‘a’ trong xâu.

SXAU2.inp	SXAU2.out
abaed	2 bed

Thuật toán:

Duyệt từ đầu xâu đến cuối xâu, nếu s[i] = ‘a’ thì tăng đếm lên. Đưa đếm ra.

Tìm kí tự ‘a’ đầu tiên, xoá đi, đến khi không còn kí tự ‘a’ trong xâu.

```
#include <bits/stdc++.h>

using namespace std;
string s;

int main()
{
   cin>>s;
   int dem = 0;
   for(int i=0;i<=s.length()-1;i++)
      if(s[i] == 'a') dem++;
   int i = 0;
   int k = 1;
   while(i <= s.length()-1)
   {
      if(s[i] == 'a') s.erase(i,1);
      else i++;
   }
   cout<<dem<<endl;
   for(int i=0;i<=s.length()-1;i++) cout<<s[i];

   return 0;
}
```

81) Xau3

Sxau3.*: Cho s, k, n.

- ✓ Xóa xâu s từ vị trí k đi n kí tự. Đưa s ra.
- ✓ Xóa k vị trí đầu tiên.

✓ Xóa N kí tự cuối cùng.

Thuật toán:

Dùng các hàm cơ bản của xâu.

SXAU3.inp	SXAU3.out
sfasecw 1 2	ssecw secw se

```
#include <bits/stdc++.h>

using namespace std;
string s;
int k,n;

int main()
{
   getline(cin,s);
   cin>>k>>n;

   s.erase(k,n);
   cout<<s<<endl;

   s.erase(0,k);
   cout<<s<<endl;

   s.erase(s.length()-n,n);
   cout<<s;

   return 0;
}
```

82) Xau4

Sxau4.*: Cho s1, s2. Xóa tất cả xâu s1 trong xâu s2.

SXAU4.inp	SXAU4.out
ab	eebbac

abeeabbbac	

Thuật toán:

Chừng nào còn xuất hiện S1 trong S2, thì xoá từ vị trí tìm thấy đi length(s1) kí tự.

Đưa xâu sau khi xử lí ra.

```
#include <bits/stdc++.h>


using namespace std;


string s1,s2;
int main()
{
   getline(cin,s1);
   getline(cin,s2);
   while(s2.find(s1) >= 0)
   {
      int p = s2.find(s1);
     if(p >= 0) s2.erase(p,s1.length());
     else break;
   }
   cout<<s2;
   return 0;
}
```

83) Xau5

Cho s. Chuẩn hóa xâu: xóa các dấu cách thừa chỉ giữ lại 1 dấu cách.

Thuật toán:

Chừng nào còn hai dấu cách đứng cạnh nhau, thì xoá đi từ vị trí tìm thấy đi 1 kí tự. Nếu kí tự đầu tiên hoặc cuối cùng là dấu cách, thì xoá dấu cách đó đi.

SXAU5.inp	SXAU5.out
thay thanh	thay thanh

```
#include <bits/stdc++.h>


using namespace std;
string s;
```

```
void chuanhoaxau()
{
   while(s.find("  ") >= 0)
   {
      int p = s.find("  ");
      if(p >= 0) s.erase(p,1);
      else break;
   }
   if(s[0] == ' ') s.erase(0,1);
   if(s[s.length()] == ' ') s.erase(s.length(),1);
}
int main()
{
   getline(cin,s);
   chuanhoaxau();
   cout<<s;
   return 0;
}
```

84) Xau6

Cho s. Chuẩn hóa xâu: cho chữ đầu tiên của xâu thành chữ hoa.

Thuật toán:

Bước 1: Chuẩn hoá xâu, xoá các dấu cách thừa đi.

Bước 2: Duyệt từ đầu xâu đến cuối xâu, nếu S[i] = ‘ ‘ (dấu cách) thì upcase s[i+1] lên.

SXAU6.inp	SXAU6.out
thay thanh	Thay thanh

```
#include <bits/stdc++.h>

using namespace std;

string s;
void chuanhoaxau()
{
   while(s.find("  ") >= 0)
```

```
    {
        int p = s.find("  ");
        if(p == -1) break;
        s.erase(p,1);
    }
    if(s[0] == ' ') s.erase(0,1);
    if(s[s.length()] == ' ') s.erase(s.length(),1);
}
char upcase(char u)
{
    if(u >= 'a' && u <= 'z') u -= 32;
    return u;
}
int main()
{
    getline(cin,s);
    chuanhoaxau();
    s[0] = upcase(s[0]);
    cout<<s;
    return 0;
}
```

85) Xau7

Cho xâu s. Chuẩn hóa xâu: cho tất cả kí tự chữ trong xâu thành chữ hoa.

Thuật toán:

Bước 1: Chuẩn hoá xâu, xoá các dấu cách thừa đi.

Bước 2: Duyệt từ đầu xâu đến cuối xâu, đến phần tử thứ I thì upcase s[i] lên.

SXAU7.inp	SXAU7.out
thay Thanh	THAY THANH

```
#include <bits/stdc++.h>


using namespace std;
string s;
void chuanhoaxau()
```

```
{
   while(s.find("  ") >= 0)
   {
      int p = s.find("  ");
      if(p == -1) break;
      s.erase(p,1);
   }
   if(s[0] == ' ') s.erase(0,1);
   if(s[s.length()] == ' ') s.erase(s.length(),1);
}
char upcase(char u)
{
   if(u >= 'a' && u <= 'z') u -= 32;
   return u;
}
int main()
{
   getline(cin,s);
   chuanhoaxau();
   for(int i=0;i<=s.length();i++)
      s[i] = upcase(s[i]);
   cout<<s;
   return 0;
}
```

86) Xau8

Cho xâu S gồm các kí tự a….z, A…Z.

Chuyển hóa xâu.

SXAU8.inp	SXAU8.out
thay thanh	Thay Thanh

```
#include <bits/stdc++.h>

using namespace std;
string s;
```

```
void chuanhoaxau()
{
   while(s.find("  ") >= 0)
   {
      int p = s.find("  ");
      if(p == -1) break;
      s.erase(p,1);
   }
   if(s[0] == ' ') s.erase(0,1);
   if(s[s.length()] == ' ') s.erase(s.length(),1);
}
int main()
{
   getline(cin,s);
   chuanhoaxau();
   for(int i=1;i<=s.length()-1;i++)
      if(s[i] >= 'A' && s[i] <= 'Z') s[i] += 32;
   if(s[0] >= 'a' && s[0] <= 'z') s[0] -= 32;
   for(int i=1;i<=s.length()-1;i++)
      if(s[i] == ' ' && s[i+1] >= 'a' && s[i+1] <= 'z') s[i+1] -= 32;
   cout<<s;
   return 0;
}
```

87) Xau9

Cho s1, s2.

Đếm số lần xuất hiện s1 trong s2.

Thuật toán:

Tìm vị trí đầu tiên S1 trong S2 lưu vào p, tăng biến đếm lên 1 đơn vị, xoá trong xâu S2 từ vị trí 1 tới P + length(s1) kí tự.

Đưa đếm ra.

SXAU9.inp	SXAU9.out
ab abcabb	2

```
#include <bits/stdc++.h>

using namespace std;
string s1,s2;
int main()
{
   getline(cin,s1);
   getline(cin,s2);
   int dem = 0;
   while(s1.find(s2) >= 0)
   {
      int p = s1.find(s2);
      if(p == -1) break;
      dem++;
      s1.erase(p,s2.length());
   }
   cout<<dem;
   return 0;
}
```

88) Xau10

Cho S. Đếm số lượng xuất hiện của các kí tự trong S dạng tăng.

Thuật toán: Đếm phân phối các kí tự.

Trong C++ ta sort xâu, nén xâu và đưa xâu nén và số lần xuất hiện ra.

QHD10.inp	QHD10.out
abcabb	a 2 b 3 c 1

```
#include <bits/stdc++.h>
#define nmax 100000
using namespace std;

string s;
char temp;
```

```
int main()
{
   getline(cin,s);
   for(int i=0;i<=s.length()-2;i++)
      for(int j=i+1;j<=s.length()-1;j++)
         if(s[i] >= s[j])
         {
            temp = s[i];
            s[i] = s[j];
            s[j] = temp;
         }
   int dem = 1;
   for(int i=0;i<=s.length()-1;i++)
      if(s[i] == s[i+1]) dem++;
      else
      {
         cout<<s[i]<<" "<<dem<<endl;
         dem = 1;
      }
   return 0;
}
```

89) Xau12

Cho xâu S.

+ Chuẩn hóa xâu
+ Tách họ và tên

QHD12.inp	QHD12.out
dang tuan thanh	Dang Tuan Thanh

Thuật toán:

Bước 1: Chuẩn hoá xâu, xoá dấu cách thừa đi.

Bước 2: Thay các kí tự đầu từ thành chữ hoa.

Bước 3: Duyệt từ cuối xâu về đầu xâu, nếu s[i] là dấu cách, thì copy trong xâu S từ vị trí i+1 đến length(s) – I kí tự.

```
#include <bits/stdc++.h>
```

```
using namespace std;
string s;
void chuanhoaxau()
{
   while(s.find("  ") >= 0)
   {
      int p = s.find("  ");
      if(p == -1) break;
      s.erase(p,1);
   }
   if(s[s.length()] == ' ') s.erase(s.length(),1);
   for(int i=0;i<=s.length()-1;i++)
      if(s[i] == ' ' && s[i+1] >= 'a' && s[i+1] <= 'z') s[i+1] -= 32;
   if(s[0] == ' ') s.erase(0,1);
   if(s[0] >= 'a' && s[0] <='z') s[0] -= 32;
}
int main()
{
   getline(cin,s);
   string s2= "";
   chuanhoaxau();
   int i = s.length()-1;
   while(s[i] != ' ')
   {
      s2 = s[i] + s2;
      s.erase(i,1);
      if(s[i] == ' ') break;
      i--;
   }
   cout<<s<<endl<<s2;
   return 0;
}
```

90) Xau13

Cho S. Đưa các số trong xâu ra

QHD13.inp	QHD13.out
Lkflfan12kajfoaj224a;lsdm53	12 224 53

Thuật toán:

Duyệt từ đầu xâu đến cuối xâu, nếu S[i] là số thì nối vào xâu tạm, ngược lại đưa xâu tạm ra và khởi tạo xâu tạm bằng rỗng.

```
#include <bits/stdc++.h>


using namespace std;
string s;
int main()
{
   getline(cin,s);
   string p = "";
   int ss=0;;
   for(int i=0;i<=s.length();i++)
      if(s[i] >= '0' && s[i] <='9') p = p + s[i];
      else
         if(p != "")
         {
            int pp = stoi(p);
            cout<<pp<<endl;
            p="";
            ss+=pp;
         }
   cout<<ss;
   return 0;
}
```

---Hết---

Hỗ trợ qua zalo: 0854518333

"Mỗi ngày tôi chọn một niềm vui,

Chọn những bông hoa, chọn những nụ cười…"

Các bạn tìm đọc tài liệu của cùng tác giả: