MỘT SỐ BÀI TẬP CÓ LỜI GIẢI CHUYÊN ĐỀ SỐ HỌC

I. HỆ ĐẾM

Bài 1: Đổi số ở hệ nhị phân sang hệ thập phân

Mã nguồn bên dưới sử dụng xâu để lưu dãy nhị phân nhập vào, và có thể đổi một số nhị phân thực sang hệ thập phân.

```
#include <iostream>
#include <cmath>

float _2to10(std::string s)
{
    char dau=0;
    if (s[0]=='-')
    {
        dau=1;
        s.erase(0,1);
    }
    short vt_cham;
    vt_cham=s.find('.');
    float t=0;
    if (vt_cham!=std::string::npos)
            s.erase(vt_cham,1);
    else
        vt_cham=s.size();
    for (int i=0;i<s.size();i++)
        t=t+(s[i]-48)*std::pow(2,vt_cham-i-1);
    if (dau==0)
        return t;
    return -t;
```

```
}


int main()
{
   std::cout<<"Chuong trinh chuyen so nhi phan sang so thap phan:\n";
   std::cout<<"Nhap so nhi phan: ";
   std::string s;
   std::cin>>s;
   std::cout<<"So thap phan tuong ung la: "<<_2to10(s);
}
```

Bài 2: Đổi số từ hệ thập phân sang hệ nhị phân

Áp dụng được với số thực có dấu trong hệ thập phân, sử dụng xâu để lưu các phần nguyên và phần phân của số tương ứng trong hệ nhị phân

```
#include <iostream>
#include <algorithm>
#include <string>


std::string doi_pn(int pn)
{
   std::string s;
   do
   {
      s.push_back(pn%2+48);
      pn=pn/2;
   }
   while (pn!=0);
   std::reverse(s.begin(),s.end());
   return s;
```

```
}
std::string doi_pp(float pp, int so_pp)
{
    std::string s;
    for (int i=0;i<so_pp;i++)
    {
        float t=pp*2;
        s.push_back(int(t)+48);
        pp=t-int(t);
    }
    return s;
}
int main()
{
    std::string s;
    std::cout<<"Nhap so o he thap phan: ";
    std::cin>>s;
    char dau='+';
    if (s[0]=='-')
    {
        dau='-';
        s.erase(0,1);
    }
    std::cout<<"So tuong ung o he nhi phan la: ";
    int pn=std::stoi(s);
    std::string sn=doi_pn(pn);
    int so_pp=5;
    int vt_cham=s.find('.');
```

```
    if (vt_cham!=std::string::npos)
        {
            float pp=stof(s)-pn;
            std::string sp=doi_pp(pp, so_pp);
            std::cout<<dau<<sn<<'.'<<sp;
        }
    else
        std::cout<<dau<<sn;
}
```

Bài 3: Đổi số từ hệ nhị phân sang hệ thập lục phân

Mã nguồn dưới đây sử dụng phương pháp đổi trực tiếp bằng cách nhóm 4 chữ số trong hệ nhị phân để chuyển sang hệ thập lục phân, sau đó ghép lại để được kết quả là số tương ứng trong hệ thập lục phân

```
#include <iostream>
#include <string>
using namespace std;

std::string s;
void ch_hoa(std::string &s)
{
        if (s.size()%4==1)
            s="000"+s;
        else if (s.size()%4==2)
            s="00"+s;
        else if (s.size()%4==3)
            s="0"+s;
}
int main()
{
```

```
 std::cout<<"Nhap so nhi phan: ";
 std::cin>>s;
 ch_hoa(s);
 unsigned i=0;
 while (i<s.size())
  {
     if (s.substr(i,4)=="0000") std::cout<<'0';
     else if (s.substr(i,4)=="0001") std::cout<<'1';
     else if (s.substr(i,4)=="0010") std::cout<<'2';
     else if (s.substr(i,4)=="0011") std::cout<<'3';
     else if (s.substr(i,4)=="0100") std::cout<<'4';
     else if (s.substr(i,4)=="0101") std::cout<<'5';
     else if (s.substr(i,4)=="0110") std::cout<<'6';
     else if (s.substr(i,4)=="0111") std::cout<<'7';
     else if (s.substr(i,4)=="1000") std::cout<<'8';
     else if (s.substr(i,4)=="1001") std::cout<<'9';
     else if (s.substr(i,4)=="1010") std::cout<<'a';
     else if (s.substr(i,4)=="1011") std::cout<<'b';
     else if (s.substr(i,4)=="1100") std::cout<<'c';
     else if (s.substr(i,4)=="1101") std::cout<<'d';
     else if (s.substr(i,4)=="1110") std::cout<<'e';
     else std::cout<<'f';
     i=i+4;
  }
}
```

Bài 4: Cho số nguyên dương *n*. Người ta phân tích *n* thành tổng các số nguyên dương theo qui tắc như sau: Nếu có thể phân tích *n* thành tổng hai số *x, y* mà hiệu của chúng đúng bằng *k* cho trước thì phân tích. Nếu không thể phân tích *n* như trên thì để nguyên *n*. Các số *x, y* đến lượt mình lại được phân tích theo qui tắc nói trên.

Hỏi cuối cùng *n* được phân tích thành tổng của bao nhiêu số hạng

Ví dụ, nếu *n*=6; *k*=2 thì đầu tiên 6=4+2. Số 2 không thể phân tích được nữa tuy nhiên số 4 lại có thể phân tích 4=3+1. Số 3 và số 1 không phân tích được nữa. Như vậy cuối cùng 6 được phân tích thành tổng của ba số (6=3+1+2).

Input*:* Hai số n, k ($n,k \leq 10^9$).

Output*:*Số lượng số thu được khi phân tích n.

Hướng dẫn: Giả sử n phân tích thành tổng của hai số nguyên x và y. Khi đó điều kiện thỏa mãn là x-y=k. Do vậy ta có hệ phương trình

$$\begin{cases} x + y = n \\ x - y = k \end{cases} => \begin{cases} x = \frac{n+k}{2} \\ y = \frac{n-k}{2} \end{cases}$$

Do vậy từ x và y ta dễ dàng tìm được số lượng số thu được khi phân tích n

Code chương trình :

```
int f(int n)
{
	if (n<=k) return 1;
if ((n+k) %2!=0) return 1;
x=(n+k)/2;
y=(n-k)/2;
return (f(x)+f(y));
}
int main()
{
cin>>n>>k;
cout<<f(n) ;
}
```

II. SỐ NGUYÊN TỐ

Bài 5: Kiểm tra tính nguyên tố của số tự nhiên n.

```
#include <iostream>
#include <ctime>
```

```
bool ktra(long long n)
   {
      if (n<2)
         return 0;
      if (n==2 || n==3)
         return 1;
      if (n%2==0 || n%3==0)
         return 0;
      for (long long k=5;k*k<=n;k+=6)
         if (n%k==0||n%(k+2)==0)
         return 0;
      return 1;
   }
int main()
{
   long long n;
   std::cout<<"Nhap n: ";
   std::cin>>n;
   std::cout<<n;
   float t=clock();
   if (ktra(n)==0)
    std::cout<<" khong phai la so nguyen to!";
   else
    std::cout<<" la so nguyen to!";
   t=(clock()-t)/1000;
    std::cout<<std::endl<<"Thoi gian thuc hien: "<<t<<"s";
}
```

Bài 6: Dùng sàng số nguyên tố để liệt kê các số nguyên tố trong [1,n]

```
#include <iostream>
#include <ctime>
#include <vector>

int main()
{
    std::ios_base::sync_with_stdio(0);
    std::cout<<"n: ";
    long long n;
    std::cin>>n;
    float t=clock();
    if (n==1)
        std::cout<<"Khong co so nguyen to nao trong [1,"<<n<<"]!";
    else
    {
        std::cout<<"Cac so nguyen to trong [1,"<<n<<"]: ";
        std::vector<bool> a(n+1,1);
        for (int i=2;i*i<=n;i++)
            if (a[i]==1)
                for (int j=i*i;j<=n;j+=i)
                    a[j]=0;
        for (int i=2;i<=n;i++)
            if (a[i]==1)
                std::cout<<i<<' ';
    }
    std::cout<<std::endl<<"Thoi gian: "<<(clock()-t)/1000<<'s';
}
```

Bài 7: Phân tích số

Cho số nguyên dương M. Hỏi có thể phân tích M thành tổng của 2 hoặc 3 số nguyên tố khác nhau được không?

Dữ liệu: vào từ tệp văn bản NGUYENTO.INP gồm 1 số nguyên dương M ($M \leq 10^4$)

Kết quả: ghi ra tệp văn bản NGUYENTO.OUT gồm 1 số là số cách phân tích N thành tổng của 2 hoặc 3 số nguyên tố khác nhau.

Ví dụ:

NGUYENTO.INP	NGUYENTO.OUT
10	2

Giải thích **:** 1. 10 = 3 + 7 và 2. 10 = 2 + 3 +5

Hướng dẫn:

+ Thuật toán 1:

- Viết 1 hàm kiểm tra tính nguyên tố của một số nguyên
- Dùng 2 vòng for để đếm xem có bao nhiêu cặp hai số có tổng = M

```
For i:=2 to m-1 do
    For j:=i+1 to m do if (NT(i)) and (NT(j)) and (i+j=M) then inc(dem);
```

- Dùng 3 vòng for lồng nhau để đếm xem có bao nhiêu cặp 3 số NT có tổng =M

* Nhận xét làm cách này sẽ chạy mất nhiều thời gian nếu M lớn

+ Thuật toán 2:

- Dùng mảng 1 chiều để lưu các số nguyên tố từ 1 đến M (có K số nguyên tố) (Có thể dùng lệnh đếm từng số hoặc dùng sàng để lấy ra các số nguyên tố)
- Dùng 2 vòng for để đếm ra kết quả:

Code chương trình :

```
void sang(int n)
{
   for (int i=1; i<=n; i++)  nt[i]=0;
   nt[1]=1;
   for (int i=2; i<=int(sqrt(n)); i++)
      if (nt[i]==0)
   {
      int j=2;
      while (i*j<=n)
```

```
      {
         nt[i*j]=1;
         j++;
      }
   }
}
int main()
{
   cin>>n;
   sang(n);
   int k=0;
   int dem=0;
   for (int i=1; i<=n; i++)
     if (nt[i]==0)    a[++k]=i;
   for (int i=1; i<=k/2; i++)
     {
       if (nt[n-a[i]]==0 && a[i]<n-a[i] ) dem++;
       for(int j=i+1; j<=k-1; j++)
          if (nt[n-a[i]-a[j]]==0&& a[j]<n-a[i]-a[j]) dem++;
   }
   cout<<dem;
   return 0;
}
```

III. ƯỚC SỐ, BỘI SỐ

Bài 8: Phân tích số tự nhiên n (n>2) thành các thừa số nguyên tố. In ra màn hình theo dạng $n=a^i*b^j...$ VD: $n=50=2*5^2$

```
#include <iostream>
```

```
int main()
{
   std::cout<<"n: ";
   int n, i=2, d;
   std::cin>>n;
   std::cout<<n<<"=";
   while (n!=1)
   {
      d=0;
      while (n%i==0)
      {
         d++;
         n=n/i;
      }
      if (d>1&&n==1)
         std::cout<<i<<"^"<<d;
      else if (d>1)
            std::cout<<i<<"^"<<d<<'*';
      else if (d==1&&n==1)
         std::cout<<i;
      else if (d==1)
         std::cout<<i<<'*';
      i++;
   }
}
```

Bài 9: Đếm và liệt kê các ước nguyên dương của số tự nhiên n.

```
#include <iostream>
```

```
int main()
{
   std::cout<<"n: ";
   int n, t, dem=0;
   std::cin>>n;
   std::cout<<"Cac uoc: 1 ";
   for (int i=2;i*i<=n;i++)
      if (n%i==0)
   {
      t=n/i;
      if (t==i)
         {
            std::cout<<i<<' '<<' ';
            dem++;
         }
      else
         {
            std::cout<<i<<' '<<t<<' ';
            dem+=2;
         }
   }
   std::cout<<n<<'\n';
   std::cout<<"So uoc: "<<dem+2;
}
```

Bài 10: Tính tổng các ước của số tự nhiên n (n>2).

```
#include <iostream>
```

```
#include<cmath>



int main()
{
    std::cout<<"n: ";
    int n, i=2, d, t=1;
    std::cin>>n;
    int n1=n;
    while (n!=1)
    {
        d=0;
        while (n%i==0)
        {
            d++;
            n=n/i;
        }
        t=t*(std::pow(i,d+1)-1)/(i-1);
        i++;
    }
    std::cout<<"Tong uoc cua "<<n1<<": "<<t;
}
```

Bài 11: Tìm ước số chung lớn nhất của hai số.

```
#include <iostream>
#include <cmath>
#include <ctime>


```

```
int main()
{
   std::cout<<"Nhap a, b: ";
   int a, b, t, d=0;
   std::cin>>a>>b;
   while (b!=0)
   {
      d++;
      t=a;
      a=b;
      b=t%b;
   }
   std::cout<<"UCLN la: "<<a;
   std::cout<<"\nSo lan thuc hien: "<<d;
}
```

Bài 12: Cho số n, tìm số nguyên dương nhỏ nhất có chính xác n ước số. Đảm bảo với n đã cho, câu trả lời không vượt quá 10^{18}.

Input: Số nguyên n (1<=n<=1000)

Output: Số nguyên dương có chính xác n ươc số

Input	Ouput
4	6
6	12

Giả sử số cần tìm được phân tích thành thừa số nguyên tố có dạng:

$p^{\alpha_1}{}_1.....p^{\alpha_k}{}_k$.

Số lượng các ước của số này là:

$(\alpha_1+1).....(\alpha_k+1)$

Do vậy, nếu số đó có thể phân tích thành dạng có 10 số nguyên tố, thì số lượng các ước của chúng sẽ là 1024 ước. Điều này chứng tỏ ta chỉ cần 10 số nguyên tố đầu tiên thôi.

Ta sẽ sử dụng phương pháp quy hoạch động:

Cho d[i][j] là số nhỏ nhất có i ước số, mà có thể được phân tích bởi j số nguyên tố đầu tiên.

Ta sẽ tìm công thức cho d[i][j] như sau:

Ta sẽ xét tất cả các lũy thừa của số nguyên tố thứ j

Nếu số nguyên tố thứ j có số mũ k, khi đó:

D[i][j]=d[i/(k+1)][j-1]* (số nguyên tố thứ j mũ k).

Chương trình viết bằng ngôn ngữ lập trình C++:

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
ll p[10] = {2,3,5,7,11,13,17,19,23,29};
ll N,ans = 1e18;
void dfs(int i,ll x,ll n){
        if(n > N) return;
        if(n == N && x < ans) ans = x;
        for(int j=1;j<=60;j++){
                if(ans<=x*p[i]) return ;
                dfs(i+1,x*=p[i],n*(j+1));
        }
}
int main(){
        cin >> N;
        dfs(0,1,1);
        cout << ans << endl;
        return 0;
}
```

Bài 13: Số phong phú

Trong số học, số phong phú là các số mà tổng các ước số của số đó (không kể chính nó) lớn hơn số đó. Ví dụ, số 12 có tổng các ước số (không kể 12) là 1 + 2 + 3 + 4 + 6 = 16 > 12. Do đó 12 là một số phong phú. Bạn hãy lập trình đếm xem có bao nhiêu số phong phú trong đoạn [L,R].

Input: File RICHNUM.INP gồm 2 số L, R ($1 \le L \le R <= 10^5$)

Output: File RICHNUM.OUT gồm 1 số nguyên duy nhất là số số phong phú trong đoạn [L, R].

Example:

RICHNUM.INP	RICHNUM.OUT

1 50	9

Giải thích: Từ 1 đến 50 có 9 số phong phú là: 12, 18, 20, 24, 30, 36, 40, 42, 48

Hướng dẫn: Để đếm số phong phú trên đoạn [l..r] ta chỉ cần so sánh tổng các ước của i lớn hơn i thì i là số phong phú và ghi nhận kết quả (res++). Để tính tổng các ước của i ta thấy nếu i là số nguyên tố thì loại i mà không cần tìm tổng các ước. Do đó việc đầu tiên ta sàng nguyên tố. sau đó chỉ đi tính tổng các ước của các số không là số nguyên tố.

Code chương trình :

```
#define maxn 100000
using namespace std;
int l,r,s[maxn+1];
int nt[maxn+1];

void sang(int x)
{
   for (int i=1; i<=x; i++) nt[i]=0;
   nt[1]=1;
   for (int i=2; i<=int(sqrt(x)); i++)
      if (nt[i]==0)
   {
      int j=2;
      while (i*j<=x)
      {
         nt[i*j]=i;
         j++;
      }
   }
}
int main()
{
```

```
    freopen("richnum.inp","r",stdin);
    freopen("richnum.out","w",stdout);
    sang(maxn);
    cin>>l>>r;
    s[2]=1; s[2]=1; s[1]=0;
    for (int i=2; i<=r; i++)
        if (nt[i]==0) s[i]=1;
      else{
        s[i]=0;
        int p=nt[i];
        while (i%p==0)
        {
            s[i]=s[i]+p;
            p=p*nt[i];
        }
        p=p/nt[i];
        int k=i/p;
        s[i]=(s[i]+1)*(s[k]+k)-i;
        }
    int res=0;
    for (int j=l; j<=r; j++)
        if (s[j]>j) res++;
    cout<<res;
    return 0;
}
```

Bài 14: Cho các số nguyên dương n, p, q, r ($n,p,q,r \leq 10^9$). Hãy đếm xem có bao nhiêu số nguyên dương trong đoạn [1,n] chia hết cho 2 trong ba số p, q, r nhưng không chia hết cho số còn lại.

Input:Gồm nhiều dòng, mỗi dòng ghi 4 số nguyên dương n, p, q, r.

Output:Mỗi dòng ghi kết quả ứng với dòng tương ứng trong input.

Hướng dẫn: Để giải bài toán trên, việc đầu tiên chúng ta phải tìm USCLN sau đó tìm BSCNN của hai số. Từ đó sẽ giải quyết được yêu cầu của bài toán.

Code chương trình :

```
int gcd(int x, int y)
{
        int  r;
do
{
r=x%y;
x=y;
y=r;
}
   while (r!=0);
return x ;
}

int bscnn (int a, int b)
{
        int u=a, v=b;
        int x=gcd(a,b);
        return u*v/x;
}
int  dem(int n; int p; int q; int r)
{
        int a,b;
```

```
a=bscnn(p,q);
b=bscnn(a,r);
return n/a-n/b;
}
int main()
{
while (scanf("%d%d%d%d",&n,&p,&q,&r)>0)
{
        t1=dem(n,p,q,r);
        t2=dem(n,p,r,q);
        t3=dem(n,q,r,p);
            cout<<t1+t2+t3<<endl;
    }
```