Descompunerea în factori primi

Problemă

Se dă un număr natural **n.** Afișați, pe câte un rând al ecranului, factorii din descompunerea lui **n** în factori primi și puterile la care aceștia apar.

Exemplu: pentru **n** = 40 se va afișa pe ecran:

2 3

5 1

Deoarece **n**=2^3*5^1.

Cum se descompune în factori primi un număr natural?

Se consideră o valoare inițială a factorului **f=2.** Se împarte **n** la **f** cât timp se poate și se contorizează numărul de repetiții; acesta fiind puterea **p** la care va apărea factorul **f** în descompunere. Apoi se incrementează valoarea factorului și se continuă algoritmul până când **n=1.**

n=40	**f=2**
n=20	**f=2**
n=10	**f=2, p=3**
n=5	**f=5, p=1**
n=1	

```
#include <iostream>
using namespace std;
int main()
{
    int n,f,p;
    cin>>n;
    f=2;
    while(n>1)
```

```
    {
        p=0;
        while(n%f==0)
        {
            n=n/f;
            p++;
        }
        if(p>0)
            cout<<f<<" "<<p<<"\n";
        f++;
    }
    return 0;
}
```

Variante de realizare în C++ a descompunerii în factori primi

Enunț variantă descompunere	Algoritm implementat în C++

Variantă iterativă: se determină mai întâi puterea la care apare 2 în descompunere, apoi se încearca fiecare valoare a lui f, **din 2 în 2, începând cu 3.**	`#include <iostream>`
	`using namespace std;`
	`int main()`
	`{`
	`int n,f,p=0;`
	`cin>>n;`
	`f=2;`
	`while(n%f==0)`
	`{`
	`n=n/f;`
	`p++;`
	`}`
	`if(p>0)`
	`cout<<f<<" "<<p<<"\n";`
	`f=3;`
	`while(n>1)`
	`{`
	`p=0;`
	`while(n%f==0)`
	`{`
	`n=n/f;`
	`p++;`
	`}`
	`if(p>0)`
	`cout<<f<<" "<<p<<"\n";`
	`f=f+2;`
	`}`
	`return 0;`
	`}`

Enunț variantă descompunere	Algoritm implementat în C++

Cu un singur while...	`#include <iostream>`
	
	`using namespace std;`
	
	`int main()`
	`{`
	`int n,f,p;`
	`cin>>n;`
	`f=2;`
	`p=0;`
	`while(n>=f)`
	`{`
	
	`if(n%f==0)`
	`{`
	`n=n/f;`
	`p++;`
	`}`
	`else`
	`{`
	`if(p>0)`
	`{`
	`cout<<f<<" "<<p<<"\n";`
	`p=0;`
	`}`
	`f++;`
	`}`
	`}`
	`if(p>0)`
	`{`
	`cout<<f<<" "<<p<<"\n";`
	`p=0;`
	`}`
	`return 0;`
	`}`

Iterativ, conform enunțului: afișați factorii care apar în descompunerea în factori primi a lui n, **fiecare factor de un număr de ori egal cu puterea la care este scris în descompunere. Exemplu: pentru** n=40 **se va afișa:** 2 2 2 5.	`#include <iostream>`
	
	`using namespace std;`
	
	`int main()`
	`{`
	`int n,f,p;`
	`cin>>n;`
	`f=2;`
	`while(n>=f)`
	`{`
	
	`if(n%f==0)`
	`{`

Enunț variantă descompunere	Algoritm implementat în C++

	n=n/f;
	cout<<f<<" ";
	}
	else
	{
	f++;
	}
	}
	return 0;
	}

Recursiv, conform enunțului: afișați factorii care apar în descompunerea în factori primi a lui n, **fiecare factor de un număr de ori egal cu puterea la care este scris în descompunere. Exemplu: pentru** n=40 **se va afișa:** 2 2 2 5.	#include <iostream>
	
	using namespace std;
	void descompunere(int n, int f)
	{
	if(n>=f)
	{
	if(n%f==0)
	{
	cout<<f<<" ";
	descompunere(n/f,f);
	}
	else
	descompunere(n,f+1);
	}
	}
	int main()
	{
	int n,f,p;
	cin>>n;
	f=2;
	descompunere(n,f);
	return 0;
	}

Recursiv după enunțul: afișați suma puterilor factorilor din descompunerea în factori primi a lui n. **Exemplu: pentru** n=40 **se va afișa 4.**	#include <iostream>
	
	using namespace std;
	int descompunere(int n, int f)
	{
	if(n>1)
	if(n%f==0)
	return
	1+descompunere(n/f,f);
	else
	return
	descompunere(n,f+1);
	else
	return 0;
	}

Enunț variantă descompunere	Algoritm implementat în C++

	int main()
	{
	int n,f,p;
	cin>>n;
	f=2;
	cout<<descompunere(n,f);
	return 0;
	}

Probleme rezolvate

1. Se dau două numere naturale nenule **a** și **b**. Să se determine numărul din intervalul [**a**,**b**] care are număr maxim de divizori. Dacă există mai multe asemenea numere, se va afișa cel mai mare dintre ele.
   Exemplu: pentru **a=4** și **b=19** se va afișa **18** deoarece este numărul cu cel mai mare număr de divizori (12 și 18 au câte 6 divizori, dar 18 este mai mare).

```
#include <iostream>
#include <cmath>
using namespace std;
int numar_divizori(int n)
{
    int p=0,f=2,nr=1;
    while(n>1)
    {
        p=0;
        while(n%f==0)
        {
            p++;
            n=n/f;
        }
        if(p>0) nr=nr*(p+1);
        f++;
    }
    return nr;
}
int main()
{
    int a,b,x,maxx=0,val,nr;
    cin>>a>>b;
```

```
    for(x=a; x<=b; x++)
    {
        nr=numar_divizori(x);
        if(nr>=maxx)
        {
            maxx=nr;
            val=x;
        }
    }
    cout<<val;
    return 0;
}
```

2. Problema **arma –** Olimpiada Județeană de Informatică, clasa a VIII-a, 2016

Enunț

În anul 2214 a izbucnit primul război interstelar. Pământul a fost atacat de către n civilizaţii extraterestre, pe care le vom numerota pentru simplicitate de la 1 la n.

Pentru a se apăra, pământenii au inventat o armă specială ce poate fi încărcată cu proiectile de diferite greutăţi, fabricate dintr-un material special denumit narun. Dacă arma este programată la nivelul p, atunci un proiectil de greutate k va ajunge exact la distanţa kp km (k la puterea p) faţă de Pământ şi dacă în acel punct se află cartierul general al unui atacator, acesta va fi distrus. De exemplu, dacă arma este programată la nivelul 2, un proiectil de greutate 10 va distruge cartierul general al extratereştrilor situat la distanţa 102 = 100 km de Pământ.

Arma poate fi încărcată cu proiectile de diferite greutăţi, dar cum narunul este un material foarte rar şi foarte scump, pământenii vor să folosească proiectile cât mai uşoare pentru a distruge cartierele generale inamice.

Cerință

Cunoscându-se **n**, numărul atacatorilor, precum şi cele **n** distanţe până la cartierele generale ale acestora, să se scrie un program care determină:

1. cantitatea minimă de narun necesară pentru a distruge toate cartierele generale inamice;
2. nivelurile la care trebuie programată arma, pentru a distruge fiecare cartier general inamic cu o cantitate minimă de narun.

Date de intrare

Fişierul de intrare **arma.in** conține pe prima linie un număr natural **c** reprezentând cerinţa care trebuie să fie rezolvată (1 sau 2). Pe cea de a doua linie se află numărul natural **n**, reprezentând numărul atacatorilor. Pe următoarele **n** linii se află **n** numere naturale, câte un număr pe o linie; pe cea de a i-a linie dintre cele **n** ($1 \le i \le n$) se află distanţa faţă de Pământ a cartierului general al celei de a i-a civilizaţii extraterestre.

Date de ieşire

Dacă cerinţa c=1, atunci pe prima linie a fişierului **arma.out** va fi scris un număr natural reprezentând cantitatea minimă de narun necesară distrugerii tuturor cartierelor generale inamice. Dacă cerinţa este c=2, atunci fişierul de ieşire **arma.out** va conţine n linii. Pe a i-a linie **(1≤i≤n)** se va scrie nivelul la care trebuie programată arma pentru a distruge cartierul general al celei de a i-a civilizaţii extraterestre.

Restricții și precizări

- **1 ≤ n ≤ 10 000**
- Distanţele până la cartierele generale inamice sunt numere naturale nenule ≤ 2.000.000.000.
- Pentru 50% dintre teste cerinţa este 1.

Exemple

arma.in	**arma.out**	**arma.in**	**arma.out**	**Explicaţíe**
1 5 100 97 625 40353607 81	122	2 5 100 97 625 40353607 81	2 1 4 9 4	Primul cartier general se poate distruge cu un proiectil de greutate 10, programat la nivelul 2, al doilea obiectiv cu un proiectil de greutate 97 programat la nivelul 1, al treilea cu un proiectil de greutate 5 programat la nivelul 4, al patrulea cu un proiectil de greutate 7 programat la nivelul 9, iar ultimul cu un proiectil de greutate 3 programat la nivelul 4. Cantitatea minimă de narun necesară este 10+97+5+7+3=122. Nivelurile sunt în ordine: 2 1 4 9 4

Timp maxim de execuţie/test: 0.6 secunde
Memorie totală disponibilă 4 MB, din care 2 MB pentru stivă
Dimensiunea maximă a sursei: 10 KB

Descrierea soluției:

Folosind ciurul lui Eratostene, se generează numerele prime până la 48.000 cu ajutorul vectorului **a,** totodată se memorează numerele prime în vectorul **prime.**

Se descompune în factori primi fiecare număr citit, folosind pentru alegerea factorilor vectorul **prime**. Se memorează, pe un rând nou în matricea **v,** pe prima coloană factorul și pe a doua puterea la care apare. Se efectuează apoi cmmdc dintre valorile memorate pe a doua coloană a matricei.

Implementarea în C++:

```
#include <iostream>
#include <fstream>
using namespace std;
ifstream fin("arma.in");
ofstream fout("arma.out");
int a[500001], prime[100001], nr;
int eratostene()
{
    int i, j;
    for(i=4; i<=48000; i=i+2)
        a[i]=1;
    nr++;
    prime[nr]=2;
    for(i=3; i<=48000; i=i+2)
    {
        if(a[i]==0)
          for(j=2; j<=48000/i; j++)
              a[i*j]=1;
         if(a[i]==0)
        {
            nr++;
            prime[nr]=i;
        }
    }
}
int v[100][3];
int main()
{
    eratostene();
    long  i, p, a, n, j, num, b, c, r;
    long long T=0;
    fin >> p;
    fin >> n;
    if(p==1)
    {
        for(i=1; i<=n; i++)
        {
            fin >> a;
            num=0;
            j=1;
            int copie=a;
            while(a>1 && j<=nr && prime[j]<=a)
            {
```

```
            if(a%prime[j]==0)
            {
                int aux=0;
                while(a%prime[j]==0)
                {
                    aux++;
                    a=a/prime[j];
                }
                num++;
                v[num][1]=prime[j];
                v[num][2]=aux;
            }
            j++;
        }
        if(num==0 || (num==1 && v[num][2]==1) || a>1)
            T=T+copie;
        else
        {
            if(num==1)   T=T+v[num][1];
            else
            {
                b=v[1][2];
                for(j=2; j<=num; j++)
                {
                    c=v[j][2];
                    r=b%c;
                    while(r)
                    {
                        b=c;
                        c=r;
                        r=b%c;
                    }
                    b=c;
                }
                if(b==1) T=T+copie;
                else
                {
                    long long  pr=1;
                    for(j=1; j<=num; j++)
                        for(int k=1; k<=v[j][2]/b; k++)
                            pr=pr*v[j][1];
                    T=T+pr;
                }
            }
        }
```

```
        }
        fout << T;
    }
    if(p==2)
     {
        for(i=1; i<=n; i++)
        {
            fin >> a;
            num=0;
            j=1;
            int copie=a;
            while(a>1 && j<=nr && prime[j]<=a)
            {
                if(a%prime[j]==0)
                {
                    int aux=0;
                    while(a%prime[j]==0)
                    {
                        aux++;
                        a=a/prime[j];
                    }
                    num++;
                    v[num][1]=prime[j];
                    v[num][2]=aux;
                }
                j++;
            }
            if(num==0 || (num==1 && v[num][2]==1) || a>1)
                fout << 1 << "\n";
            else
            {
                if(num==1)  fout << v[1][2] << "\n";
                else
                {
                    b=v[1][2];
                    for(j=2; j<=num; j++)
                    {
                        c=v[j][2];
                        r=b%c;
                        while(r)
                        {
                            b=c;
                            c=r;
                            r=b%c;
                        }
```

```
                        b=c;
                    }
                    fout << b << "\n";
                }
            }
        }
    }
    return 0;
}
```

Aplicații ale descompunerii în factori primi

Un număr natural nenul **n** mai mare decât **1** se poate scrie astfel:

$$n = f_1^{p1} * f_2^{p2} * ... * f_k^{pk}$$

Exemplu: **n=20** se scrie ca $\mathbf{2^2*5^1}$**.** Deci **n** are **k=2** factori în descompunerea în factori primi.

1. **Numărul de divizori**

   Numărul de divizori ai numărului **n** se poate afla după formula:

   $$nr = (p_1 + 1) * (p_2 + 1) * ... * (p_k + 1)$$

   Exemplu: **n=20** are (2+1)*(1+1) divizori, adică **6**: 1, 2, 4, 5, 10, 20

   Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n,f,p=0,nr=1;
    cin>>n;
    f=2;
    while(n%f==0)
    {
        n=n/f;
        p++;
    }
    if(p>0)
```

```
        nr=nr*(p+1);
    f=3;
    while(n>1)
    {
        p=0;
        while(n%f==0)
        {
            n=n/f;
            p++;
        }
        if(p>0)
            nr=nr*(p+1);
        f=f+2;
    }
    cout<<nr;
    return 0;
}
```

2. Suma divizorilor

Suma divizorilor numărului **n** se poate afla după formula:

$$S = \frac{f_1^{p1+1} - 1}{f_1 - 1} * \frac{f_2^{p2+1} - 1}{f_2 - 1} * \dots * \frac{f_k^{pk+1} - 1}{f_k - 1}$$

Exemplu: Pentru **n=20** suma divizorilor este $S = \frac{2^3-1}{2-1} * \frac{5^2-1}{5-1} = 42$

Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n,f,p=0,S=1,nr;
    cin>>n;
    f=2;
    while(n>1)
    {
        p=1;nr=0;
        while(n%f==0)
```

```
            {
                n=n/f;
                p=p*f;
                nr++;
            }
            if(nr>0)
                S=S*(p*f-1)/(f-1);
            f=f+1;
        }
        cout<<S;
        return 0;
    }
```

3. Indicatorul lui Euler

Indicatorul lui Euler sau **totient** reprezintă numărul de numere prime cu **n,** mai mici sau egale cu **n.** Acesta se calculează după formula:

$$\varphi(n) = n * \left(1 - \frac{1}{f_1}\right) * \left(1 - \frac{1}{f_2}\right) * \ldots * \left(1 - \frac{1}{f_k}\right)$$

$$sau$$

$$\varphi(n) = (f_1 - 1) * f_1^{p1-1} * (f_2 - 1) * f_2^{p2-1} * \ldots * (f_k - 1) * f_k^{pk-1}$$

Exemplu: Pentru **n=20** $\varphi(n) = 20 * \left(1 - \frac{1}{2}\right) * \left(1 - \frac{1}{5}\right) = 8$

Programul C++:

```
#include <iostream>
using namespace std;
int main()
{
    int n,f,p=0,fi=1,nr;
    cin>>n;
    f=2;
    while(n>1)
    {
        p=1;nr=0;
        while(n%f==0)
        {
```

```
            n=n/f;
            p=p*f;
            nr++;
        }
        if(nr>0)
        {
            p=p/f;
            fi=fi*(f-1)*p;
        }
        f=f+1;
    }
    cout<<fi;
    return 0;
}
```

Probleme propuse

Factorial

1. Se dă un număr natural **n.** Afișați în câte zerouri se termină **n!** (**n!=1*2*3*...*n**).
2. Se dă un număr natural **n.** Afișați ultima cifră nenulă din scrierea lui **n!**

Produsul a n numere

3. Se dă un număr natural **n (n<=1.000.000)** și **n** numere naturale nenule cu maxim 9 cifre fiecare. Afișați numărul de zerouri în care se termină produsul acestor numere.

Concurs

4. Olimpiada Națională de Informatică, clasa a 5-a, 2017 – problema **prime**

Enunț

Eu sunt fascinată de numerele prime. Consider că numerele prime sunt "scheletul" tuturor numerelor sau "atomii" acestora, pentru că orice număr natural mai mare decât 1 poate fi scris ca un produs de numere prime. Recent am aflat şi alte proprietăţi interesante legate de numerele prime, de exemplu: 1. În şirul Fibonacci există o infinitate de numere prime. Vă mai amintiţi şirul Fibonacci? 0, 1, 1, 2, 3, 5, 8, 13, ... Este şirul în care fiecare termen, exceptând primii doi, se obţine ca suma celor doi termeni care îl precedă. 2. Există numere naturale denumite „economice”. Un număr natural este economic dacă numărul de cifre necesare pentru scrierea sa este mai mare decât numărul de cifre necesare pentru scrierea descompunerii sale în factori primi (adică decât numărul de cifre necesare pentru scrierea factorilor primi şi a puterilor acestora). De exemplu 128 este economic pentru că 128 se scrie cu 3 cifre, iar descompunerea sa în factori primi se scrie cu două cifre (2 7); 4374 este economic pentru că se scrie cu 4 cifre, în timp ce

descompunerea sa în factori primi se scrie cu 3 cifre (2*37) Observaţi că atunci când un factor prim apare la puterea 1, aceasta nu este necesar să fie scrisă. 3. Multe numere naturale pot fi scrise ca sumă de două numere prime. Dar nu toate. De exemplu, 121 nu poate fi scris ca sumă de două numere prime.

Cerinţă

Scrieţi un program care citeşte numărul natural **n** şi o secvenţă de **n** numere naturale, apoi rezolvă următoarele cerinţe:

1. determină şi afişează câte dintre numerele din secvenţa dată sunt numere prime din şirul Fibonacci;
2. determină şi afişează câte dintre numerele din secvenţa dată sunt numere economice;
3. determină şi afişează câte dintre numerele din secvenţa dată nu pot fi scrise ca sumă de două numere prime.

Date de intrare

Fişierul de intrare **prime.in** conţine pe prima linie un număr natural **c** care reprezintă cerinţa (1, 2 sau 3). Pe a doua linie se află numărul natural **n**. Pe a treia linie se află o secvenţă de **n** numere naturale separate prin spaţii.

Date de ieşire

Fişierul de ieşire **prime.out** va conţine o singură linie pe care va fi scris răspunsul la cerinţa din fişierul de intrare.

Restricţii şi precizări

- **1 < n ≤ 50**
- Dacă c=1 sau c=3 numerele naturale din şir sunt mai mari decât 1 şi mai mici decât 107.
- Dacă c=2 numerele naturale din şir sunt mai mari decât 1 şi mai mici decât 1014 .
- Pentru rezolvarea corectă a cerinţei 1 se acordă 20 de puncte; pentru rezolvarea corectă a cerinţei 2 se acordă 50 de puncte, iar pentru rezolvarea corectă a cerinţei 3 se acordă 30 de puncte.

Exemple

prime.in	**prime.out**	**Explicaţie**
1 5 2 10 13 997 233	3	Cerinţa este 1. Cele 3 numere prime din şirul Fibonacci existente în secvenţă sunt 2, 13 şi 233.
2 4 128 25 4374 720	2	Cerinţa este 2. Succesiunea conţine două numere economice (128 şi 4374).
3 5 57 30 121 11 3	4	Cerinţa este 3. Sunt 4 numere naturale din secvenţă care nu pot fi scrise ca sumă de două numere prime: 57, 121, 11, 3.

Timp maxim de execuţie/test: 1.2 secunde
Memorie totală disponibilă 24 MB din care 1 MB pentru stivă
Dimensiunea maximă a sursei: 15 KB

5. Problema **grea**, autor Cristian Dospra - EMPOWERSOFT, 2017, clasa a 6-a

Enunț

Vrăjitorul Arpsod are foarte multă treabă, așa că s-a gândit să vă ocupe timpul cu o problemă foarte grea, astfel încât acesta să poată lucra liniștit la proiectele sale despre stăpânirea lumii. Acesta vă dă T numere naturale. Pentru fiecare număr A trebuie să găsiți cel mai mare K cu proprietatea că există un șir B de numere naturale, nu neapărat distincte, astfel încât:
(B1 + 1)(B2 + 1)...(BK + 1) = A.

Cerință

Arătați-i vrăjitorului că problema nu e suficient de grea pentru voi, găsind numărul K cerut într-un timp cât mai scurt, pentru fiecare din cele T numere.

Date de intrare

Fișierul grea.in va conţine pe prima linie numărul natural T, reprezentând numărul de valori date. Urmează apoi T linii. Pe fiecare linie va exista un număr A, numărul dat de Arpsod.

Date de ieșire

Fișierul grea.out, va conţine T linii. Pe fiecare linie va exista un număr K, reprezentând numărul maxim de termeni pe care îi poate avea şirul, astfel încât să respecte proprietatea cerută. Prima linie reprezintă raspunsul pentru primul număr, a doua penrtu cel de-al doilea … şamd.

Restricții și precizări

- $1 \leq T \leq 500$
- $1.000 \leq A \leq 2.000.000.000$

Exemplu

grea.in	grea.out	Explicații
1 4	2	Ne interesează rezultatul pentru un număr (4) Şirul are 2 termeni: 1 şi 1 (1 + 1)(1 + 1) = 2 * 2 = 4

6. Problema **maxD –** Olimpiada Județeană de Informatică, clasa a 9-a, 2005

Enunt

Fiind elev în clasa a IX-a, George, îşi propune să studieze capitolul divizibilitate cât mai bine. Ajungând la numărul de divizori asociat unui număr natural, constată că sunt numere într-un interval dat, cu acelaşi număr de divizori.

De exemplu, în intervalul [1, 10], 6, 8 şi 10 au acelaşi număr de divizori, egal cu 4. De asemenea, 4 şi 9 au acelaşi număr de divizori, egal cu 3 etc.

Cerință

Scrieţi un program care pentru un interval dat determină care este cel mai mic număr din interval ce are număr maxim de divizori. Dacă sunt mai multe numere cu această proprietate se cere să se numere câte sunt.

Date de intrare

Fişierul de intrare **maxd.in** conţine pe prima linie două numere a şi b separate prin spaţiu (a≤b) reprezentând extremităţile intervalului.

Date de ieşire

Fişierul de ieşire **maxd.out** va conţine pe prima linie trei numere separate prin câte un spaţiu *min nrdiv contor* cu semnificaţia:

- *min* – cea mai mică valoare din interval care are număr maxim de divizori
- *nrdiv* – numărul de divizori ai lui *min*
- *contor* – câte numere din intervalul citit mai au acelaşi număr de divizori egal cu *nrdiv*

Restricții și precizări

- $1 \leq a \leq b \leq 1000000000$
- $0 \leq b - a \leq 10000$

Exemplu

maxd.in	maxd.out	Explicații
2 10	6 4 3	6 este cel mai mic număr din interval care are maxim de divizori egal cu 4 şi sunt 3 astfel de numere 6, 8, 10.

7. Problema **miny** - Concursul National Grigore Moisil, Lugoj, 2013, autor Carmen Mincă

Enunt

Fie N un număr natural nenul şi N numere naturale nenule: x1, x2,…, xN.

Fie P produsul acestor N numere, P=x1•x2•...•xN.

Cerință

Scrieţi un program care să citească numerele N, x1, x2,…, xN şi apoi să determine:

a) cifra zecilor produsului P;

b) cel mai mic număr natural Y, pentru care există numărul natural K astfel încât Y^k=P.

Date de intrare

Fişierul de intrare **miny.in** conţine două linii. Pe prima linie este scris numărul natural N. Pe următoarea linie sunt scrise cele N numere naturale x1, x2,…, xN, separate prin câte un spaţiu.

Date de ieșire

Fișierul de ieșire **miny.out** va conține:

- pe prima linie o cifră reprezentând cifra zecilor produsului *P*;
- pe a doua linie numărul natural M de factori primi din descompunerea în factori primi a numărului *Y*;
- pe fiecare dintre următoarele *M* linii (câte o linie pentru fiecare factor prim din descompunere) câte două valori *F* şi *E*, separate printr-un singur spaţiu, *F* reprezentând factorul prim iar *E* exponentul acestui factor din descompunerea în factori primi a lui *Y*; scrierea în fişier a acestor factori primi se va face în ordinea crescătoare a valorii lor.

Restricții și precizări

- $2 \leq N \leq 50000$
- $2 \leq x_1, x_2, \ldots, x_N \leq 10000$
- pentru rezolvarea corectă a cerinței a) se acordă *20%* din punctaj iar pentru rezolvarea corectă a ambelor cerințe se acordă *100%* din punctaj.

Exemplu

miny.in	miny.out	Explicații
6 12 5 60 25 4 36	0 3 2 2 3 1 5 1	Produsul celor 6 numere este: P=12·5·60·25·4·36=12960000. Cifra zecilor lui P este 0. Se observă că există 3 valori posibile pentru Y: 12960000, 3600 şi 60 deoarece: 129600001=12960000, 36002=12960000, 604=12960000. Cea mai mică valoare dintre aceste valori este 60, astfel Y=60=22*3*5.
3 2 5 7	7 3 2 1 5 1 7 1	Produsul celor 3 numere este: P=2·5·7=70. Cifra zecilor lui P este 7. Există o singură valoare posibilă pentru Y: 70.

Diverse

8. Se dau n numere naturale şi un număr natural k. Afişaţi acele numere date care au cel puţin k divizori. (***Variantă Bacalaureat 2009***)
9. Din fișierul text **descompunere.in** se citesc maxim 1000 de numere naturale nenule, mai mari decât 1. Afișați câte dintre numerele citite au exact **k** factori în descompunerea în factori primi.
10. Din fișierul text **descompunere.in** se citesc maxim 1000 de numere naturale nenule, mai mari decât 1. Afișați câte dintre numerele citite au exact **același număr de** factori în descompunerea în factori primi.
11. Din fișierul text **descompunere.in** se citesc maxim 1000 de numere naturale nenule cu maxim 3 cifre fiecare, mai mari decât 1. Afișați, în ordine crescătoare, numerele prime.
12. Din fișierul text **descompunere.in** se citesc maxim 1000 de numere naturale nenule, mai mari decât 1. Afișați, pe câte un rând al ecranului, cel mai mic și cel mai mare divizor prim al fiecărui număr citit, separate prin câte un spațiu, pentru toate numerele care au un număr de divizori mai mare strict decât 3.