a.) Határozzuk meg kísérletileg a képernyő elektromos töltésének az előjelét. Találjunk ki több eljárást is attól függően, hogy milyen eszközökkel rendelkezünk, pl. elektroszkóp, üvegrúd, műanyag vonalzó, stb.

b.) A töltésvizsgálatot végezzük el rögtön a képernyő kikapcsolását követően is!

A feladat megoldása

a.) Egy pozitívan, papírral dörzsölt üvegrúddal feltöltött elektroszkóp gömbjét a bekapcsolt képernyőhöz közelítjük. A lemezek még jobban szétnyílnak.

Ebből, az elektrosztatikus megosztás alapján, arra következtethetünk, hogy a képernyő töltése *pozitív* előjelű.

Egy selyemcérna-szálra kis alumínium golyót kötünk. Feltöltjük ismert előjelű töltéssel, majd a TV képernyőjéhez közelítjük. Így, ha például a golyó pozitív, a képernyő eltaszítja, ebből nyilván következik, hogy a töltése *pozitív*.

b.) Hasonlóan járhatunk el a frissen kikapcsolt képernyő töltésvizsgálatánál is. Ekkor a töltése *negatív*nak adódik.

Bíró Tibor feladata

feladatmegoldók rovata

Érdekes informatika feladatok

A bolhától a királynőig

A nagyváradi Szent László Római Katolikus Líceumban megrendezett II. Nemzetközi Magyar Matematikaverseny (5–8. évfolyam) egyik 6. osztályosoknak szánt feladata a következő volt :

„Egy 5×5-ös négyzetrács bal alsó négyzetében ül egy bolha. Ugrania csak jobbra vagy felfelé szabad, de az ugrás hossza bármekkora lehet. Hányféleképpen juthat el a jobb felső négyzetbe?

Horváth Katalin, Komárom”

A megoldásban a feladat szerzője a következőket vázolta fel:

- A bal alsó sarokból indulva jobbra a második négyzetre egyféleképpen ugorhat a bolha.
- A bal alsó sarokból indulva jobbra a harmadik négyzetre kétféleképpen ugorhat a bolha: 1+1=2
- A bal alsó sarokból indulva jobbra a negyedik négyzetre 4 féleképpen ugorhat a bolha, hiszen az alsó sor bármelyik tőle balra levő négyzetéről el tud ide jutni 1+1+2 = 4. Hasonlóan az alsó sor utolsó négyzetére

1+1+2+4=8 féleképpen tud ugrani. Ugyanez érvényes az első oszlopra is.

- Nézzük például alulról a második sor negyedik négyzetét. Ide a közvetlenül alatta levő négyzetről, vagy az adott sorban tőle balra levő négyzetekről tud átugrani: 4+1+2+5=12
- Alulról a harmadik sor negyedik négyzetébe az alatta levő négyzetekről, vagy a tőle balra levő négyzetekről tud átugrani:4 + 12 + 2 + 5 + 14= 37.
- Így kitölthetjük a mellékelt négyzetrácsot, és eszerint a jobb felső sarokba 838 féleképpen ugorhat a bolha.

8	28	94	289	838
4	12	37	106	289
2	5	14	37	94
1	2	5	12	28
1	1	2	4	8

Ha megnézzük a feladatot, és a megoldást is, egyből a dinamikus programozás ugrik be. A dinamikus programozás lentről-felfelé (egyszerűtől a bonyolult felé) építkezést jelent: kiindulva a triviálisan egyszerű részfeladatok nyilvánvaló optimális megoldásaiból felépítjük, lépésről-lépésre, az egyre bonyolultabb részfeladatok optimális megoldásait, ezekből pedig végül az eredeti feladat megoldását.

Az optimum-értékeket tároló tömb triviális részfeladatokat képviselő, eleve kitöltött celláitól elindulva egyre több „szomszédos cellát" töltünk ki, míg végül ki tudjuk tölteni az eredeti feladatot képviselő cellát is.

Az előbbi feladatban a triviális részfeladatot képviselő, eleve kitöltött cella a bal alsó sarok, ahonnan a bolha indul, s nyilvánvaló, hogy oda csak egyféleképpen juthatott el, tehát az 1-es értéket tartalmazza. A táblázat (négyzetrács) többi cellájának (négyzetének) értéke kezdetben 0, és ezeket kell meghatározni.

A dinamikus programozás lényege annak az általános képletnek a megtalálása, amely matematikailag leírja az optimális építkezés módját: az optimumok tömbje valamely „apa-cellája", *mely közvetlen* „fiú-cellák" értékeitől, *milyen módon* függ?

Az alábbi táblázatokban azt jelöltük, hogy az egyes cellák értékei milyen módon függnek az előző cellák értékeitől:

1				

A triviális részfeladat eleve kitöltött cellája

1	1			

Jobbra az első cella

4	12	37		
2	5	14		
1	2	5		
1	1	2		

Egy közbelső cella

Természetesen a feladatot általánosan is megfogalmazhatjuk egy $n \times n$-es négyzetrácsra, és ha C/C++ szabvány szerint a táblázatot 0-tól n–1-ig indexeljük, a következő matematikai képletet tudjuk felírni a táblázat elemeire:

$$t_{n-1,0} = 1$$

$$\forall i = \overline{n-1,0},\ \forall j = \overline{0,n-1} : t_{i,j} = \sum_{l=n-1}^{i+1} t_{l,j} + \sum_{l=0}^{j-1} t_{i,l}$$

vagyis, ha a képletet leprogramozzuk C++-ban:

```
t[n-1][0] = 1;

for(int i=n-1;i>=0;--i)
     for(int j=0;j<n;++j)
     {
          for(int l=n-1;l>i;--l)
               t[i][j]+=t[l][j];
          for(int l=0;l<j;++l)
               t[i][j]+=t[i][l];
     }
```

A teljes feladat megoldása pedig így nézne ki:

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int n=0;
    cout<<"n = ";
    cin>>n;
    long long t[n][n];
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
           t[i][j] = 0;

    t[n-1][0] = 1;
```

```
    for(int i=n-1;i>=0;--i)
         for(int j=0;j<n;++j)
         {
              for(int l=n-1;l>i;--l)
                   t[i][j]+=t[l][j];
              for(int l=0;l<j;++l)
                   t[i][j]+=t[i][l];
         }

    for(int i=0;i<n;++i)
    {
         for(int j=0;j<n;++j)
           cout<<setw(10)<<t[i][j];
         cout<<endl;
    }

    cout<<endl<<endl;
    cout<<t[0][n-1]<<endl;
    return 0;
}
```

Egy $n \times n$-es sakktáblán a rejtőzködő fekete bástya a bal alsó sarok fekete négyzetéről indul, és felfelé, valamint jobbra léphet akárhányat, de úgy, hogy mindig csak fekete négyzetre kerülhet. Hányféleképp juthat így el a rejtőzködő bástya egy megadott x, y négyzetre a sakktáblán?

Ez a feladat egyértelműen visszavezethető a bolhás feladatra, azonban itt már minden második cellát kell csak figyelembe venni a négyzetrácsból.

A feladatot azért kellett átfogalmazni egy tetszőleges x, y négyzetre, mert páros n esetén a bástya soha nem juthat el a sakktábla jobb felső négyzetébe.

5×5-ös sakktábla esetében a megoldást a következő táblázat alapján lehet meghatározni:

2		5		14
1		2		5
1		1		2

A táblázat kitöltésének gondolatmenete ugyanaz, mint a bolhás feladatnál, viszont vigyázni kell a sakktábla bejárásával.

Észre kell vennünk, hogy másképp néz ki a sakktábla páros n-ek esetén (a legfelső sora fehér négyzettel kezdődik), és másképp néz ki páratlan n-ek esetén (a legfelső sora fekete négyzettel kezdődik), tehát a táblázat megfelelő celláinak (a sakktábla fekete négyzeteinek) a bejárása, és az összeg kiszámítása a következőképpen történik C++-ban:

```
for(int i=n-1;i>=0;--i)
    for(int j=0;j<n;++j)
    {
      if(n%2==0) h=i+j+1;
      else h=i+j;
      if(h%2==0)
      {
          for(int l=n-1;l>i;--l)
               t[i][j]+=t[l][j];
          for(int l=0;l<j;++l)
               t[i][j]+=t[i][l];
      }
    }
```

Itt a *h* érték meghatározása jött be pluszban, ez határozza meg a fekete négyzetek helyzetét a sakktáblán.

A teljes program pedig:

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int n=0;
    cout<<"n = ";
    cin>>n;

    int x=0, y=0;
    cout<<"x = ";
    cin>>x;
    cout<<"y = ";
    cin>>y;

    long long t[n][n];
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
           t[i][j] = 0;

    t[n-1][0] = 1;
    int h=0;

    for(int i=n-1;i>=0;--i)
        for(int j=0;j<n;++j)
        {
          if(n%2==0) h=i+j+1;
          else h=i+j;
          if(h%2==0)
          {
              for(int l=n-1;l>i;--l)
                   t[i][j]+=t[l][j];
              for(int l=0;l<j;++l)
                   t[i][j]+=t[i][l];
          }
```

```
        }

    for(int i=0;i<n;++i)
    {
        for(int j=0;j<n;++j)
          cout<<setw(4)<<t[i][j];
        cout<<endl;
    }

    cout<<endl<<endl;
    cout<<t[y-1][x-1]<<endl;
    return 0;
}
```

Ez volt a feladat egyik általánosítási lehetősége. Egy másik pedig az lehet, ha megvizsgáljuk, mi történik akkor, ha a bolha az átlón is tud jobbra-felfele ugrani, vagyis:

> Egy $n \times n$-es négyzetrács bal alsó négyzetében ül egy bolha. Ugrania csak jobbra, felfelé, vagy átlósan jobbra-felfelé szabad, de az ugrás hossza bármekkora lehet. Hányféleképpen juthat el a jobb felső négyzetbe?

Nyilván, ebben az esetben annyi változik, hogy az összeg kiszámításánál figyelembe kell venni az adott négyzet alatti mellékátlón lévő értékeket is, a programrész tehát így módosul:

```
for(int i=n-1;i>=0;--i)
    for(int j=0;j<n;++j)
    {
        for(int l=n-1;l>i;--l)
            t[i][j]+=t[l][j];
        for(int l=0;l<j;++l)
            t[i][j]+=t[i][l];
        for(int l=j-1,k=i+1;l>=0&&k<n;--l,++k)
            t[i][j]+=t[k][l];
    }
```

5×5-ös négyzetrács esetében (ahogy az eredeti feladatnál volt), a táblázat így alakul:

8	40	158	543	1712
4	17	60	188	543
2	7	22	60	158
1	3	7	17	40
1	1	2	4	8

Megfigyelhető, míg az eredeti feladatban csupán 838-féleképp juthatott el a bolha a jobb alsó sarokból a bal felső sarokba, az átlós ugrás bevezetése miatt a lehetőségek száma itt már 1712-re emelkedett.

A teljes program a következő:

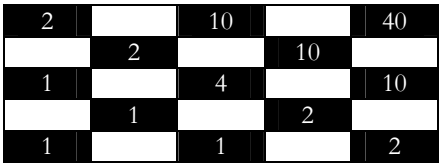
```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int n=0;
    cout<<"n = ";
    cin>>n;
    long long t[n][n];
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
           t[i][j] = 0;

    t[n-1][0] = 1;

    for(int i=n-1;i>=0;--i)
         for(int j=0;j<n;++j)
         {
              for(int l=n-1;l>i;--l)
                   t[i][j]+=t[l][j];
              for(int l=0;l<j;++l)
                   t[i][j]+=t[i][l];
              for(int l=j-1,k=i+1;l>=0&&k<n;--l,++k)
                   t[i][j]+=t[k][l];
         }

    for(int i=0;i<n;++i)
    {
         for(int j=0;j<n;++j)
           cout<<setw(9)<<t[i][j];
         cout<<endl;
    }

    cout<<endl<<endl;
    cout<<t[0][n-1]<<endl;
    return 0;
}
```

Végül hozzuk össze a két feladatot, és fogalmazzuk át a rejtőzködő bástyás feladatot rejtőzködő vezérre!

Egy $n \times n$-es sakktáblán a rejtőzködő fekete vezér a bal alsó sarok fekete négyzetéről indul, és felfelé, jobbra, valamint átlósan jobbra-felfelé léphet akárhányat, de úgy, hogy mindig csak fekete négyzetre kerülhet. Hányféleképp juthat így el a rejtőzködő vezér egy megadott x, y négyzetre a sakktáblán?

Természetesen a feladat megoldása az átlósan is ugorható bolha képletének alkalmazása a rejtőzködő bástyánál kigenerált sakktáblára!

Nézzük meg, hogyan alakul ez egy 5×5-ös sakktábla esetében:



A teljes C++ program pedig:

```
#include <iostream>
#include <iomanip>

using namespace std;

int main()
{
    int n=0;
    cout<<"n = ";
    cin>>n;

    int x=0, y=0;
    cout<<"x = ";
    cin>>x;
    cout<<"y = ";
    cin>>y;

    long long t[n][n];
    for(int i=0;i<n;++i)
        for(int j=0;j<n;++j)
           t[i][j] = 0;

    t[n-1][0] = 1;
    int h=0;

    for(int i=n-1;i>=0;--i)
        for(int j=0;j<n;++j)
        {
          if(n%2==0) h=i+j+1;
          else h=i+j;
          if(h%2==0)
          {
              for(int l=n-1;l>i;--l)
                   t[i][j]+=t[l][j];
              for(int l=0;l<j;++l)
                   t[i][j]+=t[i][l];
              for(int l=j-1,k=i+1;l>=0&&k<n;--l,++k)
                   t[i][j]+=t[k][l];
          }
        }
```

```
    for(int i=0;i<n;++i)
    {
        for(int j=0;j<n;++j)
          cout<<setw(6)<<t[i][j];
        cout<<endl;
    }

    cout<<endl<<endl;
    cout<<t[y-1][x-1]<<endl;
    return 0;
}
```

Kovács Lehel István

Kémia

K. 814. (*Hevesy György Kémiaverseny, 8.osztály*) Sokáig azt gondolták, hogy a nemesgázok teljesen reakcióképtelenek. Ezt az elképzelést a vegyészeknek sikerült megdönteniük azzal, hogy előállították több nemesgáz fluorral, illetve oxigénnel alkotott vegyületét. A nemesgázok vegyértéke 2, 4, 6 vagy 8 lehet. Egy ilyen vegyület az egyik nemesgáz oxidja is, amelyet több mint 50 éve állítottak elő. A vegyület –35,9 °C alatt sárga színű, kristályos anyag. –35,9 °C felett nagyon instabillá válik, és elemeire bomlik. 1,172 g vegyületből 20 millimol oxigéngáz szabadul fel. Melyik nemesgáz oxidjáról van szó, és mi a képlete?

K. 815. Egy szerves vegyület szenet, oxigént és hidrogént tartalmaz. A vegyületben az oxigénatomok száma fele a szénatomok számának, a szénatomok száma pedig fele a hidrogénatomok számának. 1mol O_2 gáz a vegyület 0,20 móljának tökéletes égéséhez éppen elegendő. Mi a vegyület összegképlete? Írj fel legalább egy olyan molekula szekezeti képletet, amely az adott összegképletnek megfelel.

K. 816. Elkészítünk három NaCl-oldatot, amelyek tömegszázalékban kifejezett koncentrációi egymástól rendre ugyanannyi tömeg%-ban térnek el. A három oldatból azonos tömegű részleteket összeöntünk. Mennyi lesz az így kapott oldat koncentrációja tömegszázalékban kifejezve

K. 817. Az 1,18 g/cm^3 sűrűségű 20 ^{o}C hőmérsékleten telített vizes oldat 1,00 dm^3-e 72,0 g $Ba(OH)_2 \cdot 8H_2O$ oldásával készült.

(a) Hány gramm vízmentes bárium-hidroxidot old 20 ^{o}C-on 100 g víz?

(b) Milyen tömegarányban kell fémbáriumot és vizet elegyíteni, hogy az oldat OH^--ion koncentrációja 10^{-1} mol/dm^3 legyen? Az oldat sűrűsége 1,05 g/cm^3.

K. 818. Egy szürke porkeverék lítium-alumínium-hidridet ($LiAlH_4$) és elemi alumíniumot tartalmaz. A keverék 73,0 mg-jához 200,0 mg vizet adunk. Heves gázfejlődés játszódik le, a fejlődött gáz térfogata 298 K-en és 101325 Pa nyomáson (standard állapot) 103,1 cm^3 lesz, és a reakcióban visszamaradó oldat és szilárd anyag együttes tömege 264,5 mg. A reakció után a szilárd anyagot is tartalmazó oldathoz 60,0 mg szilárd NaOH-ot adnak, és ennek hatására még 44,9 cm^3, az előzővel azonos állapotú gáz keletkezik, a visszamaradó oldat tömege pedig 320,8 mg lesz. Milyen gáz fejlődik az egyes lépésekben? Mi a lezajló kémiai reakciók egyenlete? Mi volt az eredeti porkeverék összetétele?