Државно такмичење из програмирања
Београд – 20. април 2013 .
II категорија (7. и 8. разред)

1. На већини данашњих мобилних телефона аларм може бити подешен тако да се оглашава на два начина: звони сваког дана у исто време или звони у одређено време одређеног дана. Могуће је подесити неколико аларма на мобилном телефону. Напишите програм TELEFON који за задато текуће време на основу задате информацију о сатници аларма исписује када ће се аларм следећи пут огласити звоном.

Улазни подаци. Стандардни улаз садржи садржи неколико линија. У првој линији стандардног улаза задата су три цела броја која описују текуће време: редни број дана у недељи (од 1 до 7), час и минут. У другој линији улаза задат је број аларма, цео број N (0<*N* <=100). У следећих N линија налазе се информације о подешавању ових N аларма. Опис сваког аларма састоји се од три броја: редни број дана у недељи (од 1 до 7 за дане редом понедељак,..., недеља и 0 ако аларм треба да звони сваког дана), часови (од 0 до 23), минути (од 0 до 59).

Излазни подаци. Стандардни излаз треба да садржи једну линију и три броја раздвојена размаком који редом представљају редни број дана у недељи, час и минут следећег оглашавања аларма звоном.

Пример 1		**Пример 2**	
Улаз	**Излаз**	**Улаз**	**Излаз**
2 10 20 2 1 23 15 0 10 10	3 10 10	7 1 1 3 7 0 59 7 23 59 7 1 1	7 1 1

2. На располагању нам је вага са два таса за мерење масе и тегови чије су масе 1, 3, 9, 27, ..., 3^{n-1} kg (по један примерак сваког таквог тега). На левом тасу налази се предмет масе m kg. Напишите програм VAGA који ће расподелити тегове на оба таса тако да вага буде у равнотежи. Није потребно да користите све тегове.

Улазни подаци. Стандардни улаз садржи садржи једну линију са два цела броја n ($1 \le n \le 11$) и m ($1 \le m \le 3^{n-1}$)

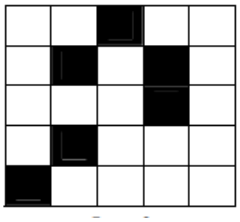
Излазни подаци. Стандардни излаз треба да садржи две линије. Ваш програм треба да испише у првом реду стандардног излаза најпре број m, а потом масе тегова са левог таса уравнотежене ваге у растућем редоследу. У другом реду потребно је исписати масе тегова са десног таса, такође у растућем редоследу.

Пример

Улаз	Излаз
4 5	5 1 3 9

3. Васа и Петар су играли игру померања пешака на квадратној табли подељеној на n^2 поља. Поља су по врстама нумерисана бројевима од 1 до n^2 (сл. 1). Поља могу бити бела и црна, али за разлику од шаха, црна поља су произвољно распоређена.

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25



Пешак се креће само по белим пољима, при чему може са текућег поља да се помери у једном потезу на друго поље само ако се та два поља налазе на истој хоризонталној или вертикалној линији и између њих не постоје црна поља. На пример, на слици 2 са поља 13 може да се у једном потезу оде на поља 8, 11, 12, 18, 23.

Написати програм KORACI који за задату величину и боје табле исписује минимални број потеза да се пешак помери са једног задатог поља на друго задато поље.

Улазни подаци. Стандардни улаз садржи садржи две линије. У првој линији су дата три цела броја N (0 < N <100), X, Y, који представљају број врста квадратне табле и нумерацију почетног поља са ког креће пешак и нумерацију крајњег поља померања пешака. У другој линији задат је број црних поља B и нумерација црних поља.

Излазни подаци. Стандардни излаз треба да садржи једну линију. Ваш програм треба да испише минималан број потеза да пешак стигне са поља X на поље Y придржавајући се правила игре. Ако није могуће да пешак стигне са поља X на поље Y, исписати –1.

Пример

Улаз	Излаз
5 2 4 6 3 7 9 14 17 21	7

Решење:

1.

```
#include <iostream>

using namespace std;

int convert_time(int d,int h,int m)
{
 return ((d-1)*24*60+h*60+m);
}


int main()
{
 int n,cl_time,current_time,current_day,time,d,h,m,i;
 cin>>d>>h>>m;
 current_day=d;
 current_time=convert_time(d,h,m);
 cl_time=100000000;
 cin>>n;
 for(i=1;i<=n;i++)
 {
   cin>>d>>h>>m;
   if(d==0)
   {
     time=convert_time(current_day,h,m);
     if (current_time>time) time=time+24*60;
     if ((time-current_time)<(cl_time-current_time))
       cl_time=time;
   }
   else
   {
     time=convert_time(d,h,m);
     if(current_time>time)time=time+24*7*60;
     if ((time-current_time)<(cl_time-current_time))
       cl_time=time;
   }
 }
 d=(cl_time/(24*60))%7+1;
 h=(cl_time%(24*60))/60;
 m=cl_time%60;
 cout<<d<<' '<<h<<' '<<m<<endl;
 return 0;
}
```

2.

```
#include <iostream>
using namespace std;
int main()
{
   long n, i=0, d[11];
   long m, x;
   cin>>n>>m;
   x=m;
   while (x>0)
   {
        d[i]=x%3; if (d[i]==2)d[i]=-1;
```

```
        x=(x-d[i])/3; i++;
    }
    cout<<m;
    x=1;
    for (int i=0;i<n;i++)
    {
        if (d[i]==-1)cout<<" "<<x;
        x*=3;
    }
    cout<<endl;
    x=1;
    for (int i=0;i<n;i++)
    {
        if (d[i]==1)
          if (i<n-1)cout<<x<<" ";
          else cout<<x;
        x*=3;
    }
    cout<<endl;
    return 0;
}
```

3.

```
#include <cstdio>
#include <queue>
using namespace std;

int a[10000];
int d[10000];
int n;

queue<int> q;


int getRow(int z)
{ return 1+(z-1)/n; }

int main()
{ int x,y,b;
  scanf("%d%d%d",&n,&x,&y);

  scanf("%d",&b);
  for(int i=1; i<=b; i++)
  { int z;
    scanf("%d",&z);
    a[z] = 1;
  }

  for(int i=1; i<=n*n; i++)
    d[i] = -1;

  d[x] = 0;
  q.push(x);
```

```
  while(!q.empty())
  { int z = q.front();
    q.pop();

    int p;
    p = z-1;
    while(getRow(p)==getRow(z) && a[p]==0)
    { if(d[p]==-1) { d[p]=d[z]+1; q.push(p); }
      p--;
    }

    p = z+1;
    while(getRow(p)==getRow(z) && a[p]==0)
    { if(d[p]==-1) { d[p]=d[z]+1; q.push(p); }
      p++;
    }

    p = z-n;
    while(p>=0 && a[p]==0)
    { if(d[p]==-1) { d[p]=d[z]+1; q.push(p); }
      p = p-n;
    }

    p = z+n;
    while(p<=n*n && a[p]==0)
    { if(d[p]==-1) { d[p]=d[z]+1; q.push(p); }
      p = p+n;
    }
  }

  printf("%d\n",d[y]);

  return 0;
}
```