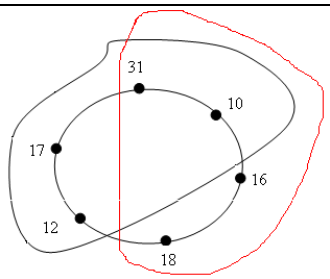
Државно такмичење из програмирања, Београд – 26. април 2014 .
I категорија (5. и 6. разред)

1. У продавници слаткиша, сваки од *N* производа који се продаје је распоређен тако да образује круг. Када купујете неколико слаткиша у овој необичној продавници, онда морате увек изабрати слаткише који стоје један поред другог у кругу. На каси продавнице, морате дати тачан износ када купујете слаткише. Цена сваког слаткиша је позитиван цео број. Ви као купац располажете с неограничено много новчаница једнаке вредности, тако да свака новчаница вреди *d* динара. Напишите програм, конзолну апликацију, **SLATKISI**, која израчунава максималан број слаткиша који можете да купите, према горе описаним условима (купујући слаткише са суседних позиција у кругу и дајући тачан износ на каси).

Улазни подаци. Стандардни улаз садржи садржи две линија. Ваш програм треба да из првог реда стандардног улаза учита два цела број раздвојених једним бланко карактером: *N*(1≤ *N* ≤ 1000*)* – број слаткиша у продавници и *d* (1≤ *d* ≤ 1 000 000 000)– вредност једне од новчаница које купац поседује. У другом реду стандардног улаза дато је *N* целих позитивних бројева, раздвојених са по једним бланком карактером који представљају цену слаткиша. Цене слаткиша су дате у смеру кретања казаљке на сату. Цена једног слаткиша је не мања од 1 и не већа од 1 000 000 000 динара.

Излазни подаци. Стандардни излаз треба да садржи једну линију. Ваш програм треба да испише један цео број: максимални број слаткиша који се може купити.

Пример

Улаз	Излаз		
6 5 31 10 16 18 12 17	4		ОБЈАШЊЕЊЕ: Можете купити највише 4 слаткиша чија сума цена је дељива са 5 – слаткиши чија је цена 12,17, 31, 10 динара (12+17+31+10=70) или 31,10,16,18 динара (31+10+16+18=75). Можете купити и слаткише чија је цена 12 и 18 динара или само слаткиш чија је цена 10 динара, али у оба случаја број купљених слаткиша је мањи од 4.

Решење:

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

int main()
{
 int cena[1001];    // cene slatkisa
 int n,d;        // n-broj slatkisa, d-vrednost jedne novcanice
 long long s;     // s-suma cena slatkisa, pocev od i-tog slatkisa
 int br, maxbr=0;   // br-broj slatkisa koji su skupljeni
 // u potencijalnoj kupovini pocev od i-tog slatkisa
 // maxbr - max broj kupljenih susednih slatkisa
 int i,j;

 cin >> n >> d;
 for (i=1;i<=n;i++)
  cin >> cena[i];
 for (i=1;i<=n;i++)
 {
  s=0;     // pocetak stvaranja nove grupe
   //kupljenih susednih slatkisa pocev od i-tog slatkisa
  br=0;
  for (j=i;j<=i+n-1;j++)  //uvrstiti po jedan slatkis dok ne testiramo sve slatkise
  {
   br++;
   if (j<=n)      //testiramo da li smo dosli do kraja kruga
    s=s+cena[j];
   else
    s=s+cena[j-n]; //slatkisi mogu biti kupljeni sa kraja i pocetka kruga
```

```
 //ako suma cena formirane grupe slatkisa je
 //deljiva sa d, onda se mogu kupiti slatkisi iz te grupe
  if ((s%d==0) && (br>maxbr))
      maxbr=br;  //novi maskimalni broj grupe slatkisa koji se mogu kupiti

    }
  }

  cout << maxbr << endl;
  return 0;
}
```

2. Неколико округа у Србији организује прославу под називом *Дани шљива*. У једној таквој општини у главној улици постоји *n* стабала шљива поређаних у облику праве линије тако да су свака два суседна стабла на различитом растојању које износи *d*. Напишите програм, конзолну апликацију, **SLJIVE**, који ће израчунати најмањи број стабала шљива које треба посадити међу постојећим стаблима у главној улици (у облику праве линије) тако да растојање између свака два суседна стабла буде једнако.
Улазни подаци. Стандардни улаз садржи садржи две линије. У првој линији стандардног улаза задат је један цео број *n* – број стабала, 0 < *n* < 1001. У другој линији улаза задато је (*n*-1) различитих целих бројева – растојања између свака два суседна стабла поређаних у облику праве линије. Бројеви су раздвојени са по једним бланко карактером и за сваки број који представља растојање важи 0 < *d* < 1 000 000 000.
Излазни подаци. Стандардни излаз треба да садржи један цео број: минималан број шљива које треба посадити у облику праве линије.

Пример: **Улаз**	**Излаз**
4 63 18 54	12

Објашњење примера: Између 1. и 2. стабла поставити 6 стабала, између 2. и 3. стабла поставити 1 стабло, између 3. и 4. стабла поставити 5 стабала.

Решење:
Стабла шљиве треба посадити тако да буду на једнаким растојањима. Дакле, растојање међу стаблима треба да дели сваки од датих растојања *d*. Број стабала ће бити мањи ако је то једнако растојање међу њима што је могуће веће. Дакле, задатак налажења минималног број стабала шљива које треба посадити у облику праве линије се своди на тражење највећег заједничког делиоца НЗД за датих (*n*-1) бројева. Након проналаска НЗД, да би добили минималан број нових стабала израчунамо количник сваког од различитих растојања и НЗД и количник умањимо за један (на пример, број нових стабала на удаљености од 18м је 1, тј. на сваких 9м поставимо 1 стабло, тј. 18:НЗД(63,18,54)-1).
Коначан резултат добијамо када саберемо број нових стабала између свака два суседна стара стабла.

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

//najveci zajednicki delilac za a, b
long long nzd(long long a, long long b)
{long long r;
 while (b!=0)
    {r=a%b;
     a=b;
     b=r;
    }
 return a;
}
```

```
int main()
{long long suma;
 int i,n,d,rastojanje;
 cin>>n>>d;
 suma=d;
 for (i=2;i<n;i++)
    {cin>>rastojanje;
     d=nzd(rastojanje,d);
     suma=suma+rastojanje;
    }
 cout<<suma/d-(n-1)<<endl;
}
```

3. Нека је дат цео позитиван број *N*. Нека *s*(*N*) означава збир цифара броја *N* (на пример, *s*(293)=2+9+3=14). Број *N* је *радостан* ако важи да *s*(*N*) = *s*(9×*N*). На пример, број 27 је радостан јер, *s*(27)=2+7=9,*s*(9×27)=*s*(243)=2+4+3=9. Број 34 није радостан, јер *s*(34)=3+4=7, а *s*(9×34)=*s*(306)=3+0+6=9≠7. Напишете програм, конзолну апликацију RADOST, који пребројава колико има радосних бројева у датом интервалу [*a*, *b*].

Улазни подаци. Учитавају се два цела позитивна броја a, b у првом реда стандардног улаза, раздвојени бланко карактером. Бројеви ***a*** и ***b*** немају више од 18 цифара и важи да $a \le b$ и $b - a \le 20\,000\,000$.

Излазни подаци. Стандардни излаз треба да садржи број радосних бројева ***x*** тако да $a \le x \le b$.

Пример

Улаз	Излаз
13 99	6

Решење:

„Радосни“ бројеви између 13 и 99 (укључено) су 18, 27, 36, 45, 90 и 99. Уочавамо и на датом тест примеру да су решења бројеви дељиви са 9.

Заиста, како је број 9×*N* дељив са 9, онда му је према правилима дељивости са 9, и сума цифара *s*(9×*N*) дељива са 9. Ако важи да *s*(*N*) = *s*(9×*N*), онда је и *s*(*N*) дељива са 9, односно 9|N.

На основу тог закључка можемо убрзати потрагу радосних бројева на датом сегменту [*a*, *b*].

Додатно, такмичари су могли претрагу убрзати и добити још ефикасније решење ако промишљено рачунамо суму цифара бројева .

Ако број *N* има суму цифара *s*, следећи могућ радостан број је t = N +9, чија сума цифара је :

s + 9 , уколико је 0 последња цифра броја *N*;

у супротном, почев од претпоследње цифре броја *N* ка цифрама на старијим позицијама док имамо узастопне деветке смањујемо *s* за 9. Заустављамо се код прве цифру различите од 9 (или ако немамо више цифара).

```
#include <iostream>
using namespace std;
int main()
{long long a,b;
 int br=0,s1,s2; //s1 je s(N), s2 je s(9*N), br je broj radosnih brojeva iz segmenta [a,b]
 long long t,i,i9; //t-pomocna promenljiva
 //i, i9 su moguci radosni brojevi iz segmenta [a,b]
 cin>>a>>b;

 //prvi broj veci od a i deljiv sa 9
 if (a%9!=0) a=9*(a/9+1);
 s1=s2=0;
 i=t=a;

 //suma cifara broja i
 while (t!=0)
 {s1=s1+t%10;
  t=t/10;
```

```
 }

 i9=t=9*a;//i9 je 9*i
 //suma cifara broja i9
 while (t!=0)
 {s2=s2+t%10;
  t=t/10;
 }

 if (a<=b)//ako je nakon operacije a=9*(a/9+1)
        //jos uvek a<=b
  do
  {if (s1==s2) br++;// nadjen radostan broj
   //priprema za narednu sumu s1
   if (i%10==0) s1=s1+9;
   else
   {t=i/10;
    while(t%10==9){s1=s1-9;t=t/10;}
   }
   //priprema za narednu sumu s2
   if (i9%10==9) s2=s2-9;
   if (i9%100<19) s2=s2+9;
   else
   {t=i9/100;
    while (t%10==9) {s2=s2-9;t=t/10;}
   }
   i=i+9;//naredni kandidat
   if (i>b) break;
   i9=i9+81;
  }while (true);
 cout<<br<<endl;
}
```