MATHEMATICAL GRAMMAR SCHOOL CUP
June, 30, 2016

SOLUTIONS

TASK 1. PIN

Let us consider unoriented graph in which the vertices are given segment (including conductor). Two pins are adjacent when they have a common point. In so defined graph, You need to count the number of vertices in the graph connected component having "special" pin. These vertices represent "electricity." The other vertices are those with "no electricity".

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

struct Point
{ int x, y; };

const int NMAX = 1024;

Point A[NMAX], B[NMAX];
int n;

bool adj[NMAX][NMAX];
bool visited[NMAX];

int compSize;

int direction(Point A, Point B, Point C)
{ int a1=B.x-A.x, a2=B.y-A.y;
  int b1=C.x-A.x, b2=C.y-A.y;
  int p=a1*b2, q=a2*b1;
  if(p>q) return +1;
  if(p<q) return -1;
  return 0;
}

bool onSegment(Point A,Point B,Point C)
{ return min(A.x,B.x)<=C.x && C.x<=max(A.x,B.x) &&
         min(A.y,B.y)<=C.y && C.y<=max(A.y,B.y);
}

bool intersect(Point A, Point B, Point C, Point D)
{ int d1 = direction(A,B,C);
  int d2 = direction(A,B,D);
  int d3 = direction(C,D,A);
  int d4 = direction(C,D,B);
  if (d1*d2<0 && d3*d4<0)return true;
  if (d1==0 && onSegment(A,B,C)) return true;
  if (d2==0 && onSegment(A,B,D)) return true;
  if (d3==0 && onSegment(C,D,A)) return true;
  if (d4==0 && onSegment(C,D,B)) return true;
  return false;
}

void dfs(int i)
{ compSize++;
  visited[i]=true;
  for(int j=0; j<n; j++)
    if(adj[i][j] & !visited[j]) dfs(j);
}

int main()
{ int i;
  cin >> n;
  for(i=0;i<n;i++)
    cin >> A[i].x >> A[i].y >> B[i].x >> B[i].y;


  for(int i=0; i<n; i++)
    for(int j=i+1; j<n; j++)
            adj[i][j] = adj[j][i] = intersect(A[i],B[i],A[j],B[j]);

  compSize=0;
  dfs(n-1);

  cout << n-compSize << endl;

  return 0;
}
```

MATHEMATICAL GRAMMAR SCHOOL CUP
June, 30, 2016

TASK 2. SEPARATION

It is important (enough) to find the solution via integer number for the system:

$$\left|\begin{array}{l} ax+by+cz=n \\ x+y+z=m \\ x\geq 0, y\geq 0, z\geq 0 \end{array}\right.$$

If we put second equation into the first (multiply second equation with a and subtract from the first):
$(b-a)y+(c-a)z=n-am$.

Let us introduce new variables: $b_1=b-a$, $c_1=c-a$, $n_1=n-am$.

If we follow constraints, than we get $0<b_1<c_1$ and $n_1>0$. So, we need to solve Diophantine equation $b_1y+c_1z=n_1$. Let us create solution in the following order (GCD=greatest common divisor):

1. Find GCD(b_1,c_1) using Euclid algorithm.
2. If n_1, is not divisible by GCD(b_1,c_1), than there is no solution for Diophantine equation. And, there isn't solution for the system, also. In that case, final result is 0.
3. If n_1, is divisible by GCD(b_1,c_1), let us divide b_1, c_1 and n_1 with GCD(b1,c1). Let us introduce unknowns $b_2=\frac{b_1}{GCD(b_1,c_1)}$, $c_2=\frac{c_1}{GCD(b_1,c_1)}$ and $n_2=\frac{n_1}{GCD(b_1,c_1)}$

   So, our Diophantine equation becomes $b_2y+c_2z=n_2$, and now it is certain that GCD(b_2,c_2)=1.
4. Using the extended Euclidean algorithm, we find a pair of integers (y',z'), and they are solutions for the Diophantine equation $b_2y+c_2z=1$ (1= GCD(b_2,c_2)). It means that $b_2y'+c_2z'=1$.
5. In the last equality we can multiply both sides with n_2, and get $b_2(n_2.y')+c_2(n_2.z')=n_2$, and that means that we have found a pair of numbers $(y_0,z_0)=(n.y',n.z')$, which are the solution of equation $b_2y+c_2z=n_2$. Then, all its solutions are set by equalities $y=y_0-c_2t$, $z=z_0+b_2t$, $t\in Z$.

The initial system becomes

$$\left|\begin{array}{l} x=m-y_0-z_0+(c_2-b_2)t \\ y=y_0-c_2t \\ z=z_0+b_2t \\ x\geq 0, y\geq 0, z\geq 0, t\in Z \end{array}\right.$$

The number of its solutions is equal to the number of integer solutions of the system of inequalities

$$\left| \begin{array}{l} t \geq \dfrac{y_0 + z_0 - m}{c_2 - b_2} \\ t \leq \dfrac{y_0}{c_2} \\ t \geq -\dfrac{z_0}{b_2} \end{array} \right.$$

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

long long euclid(long long b, long long c)
{
   while(b!=0)
   {
      c=c%b;
      swap(b,c);
   }
   return c;
}

void extended_euclid(long long b, long long c, long long & y1, long long &z1)
{
   long long q, r;
   long long z2=0, y2=1, z11, y11;
   z1=1; y1=0;
   while(b!=0)
   {
      q=c/b; r=c%b;
      c=b; b=r;
      z11=z1; y11=y1;
      z1=z2; y1=y2;
      z2=z11-z2*q;
      y2=y11-y2*q;
   }
   return;
}

int main()
{
   long long a, b, c, m, n, z, y, gcd, y0, z0;
   double p, q, p1, p2;
   cin >> a >> b >> c >> m >> n;
   b=b-a; c=c-a; n=n-a*m;
   gcd=euclid(b,c);
   if(n%gcd != 0)
   {
      cout << 0 << endl;
```

```
        return 0;
    }
    b=b/gcd; c=c/gcd; n=n/gcd;
    extended_euclid(b,c,y,z);
    y0=y*n; z0=z*n;
    p1 = double(-z0)/b;
    q = double(y0)/c;
    p2 = double(y0+z0-m)/(c-b);
    if(p1<p2) p=p2;
    else p = p1;
    p = ceil(p); q = floor(q);
    if (p<=q) cout << (long long)(q-p+1) << endl;
    else cout << 0 << endl;
    return 0;
}
```

MATHEMATICAL GRAMMAR SCHOOL CUP
June, 30, 2016

TASK 3. STUTTER

In our task, as number N is even, we have at the input (N-2) / 2 pairs of identical numbers and 2 different numbers. We want to find out those two numbers with complexity O (N).
Trivial solution of the problem is to sort the array and thus find those numbers with complexity O (N). But, because N<1000000, it is hard to satisfy memory limit in that case.

Solution for 20 pts without sorting

```
#include <cstdio>

int N;
int cnt[100005];

int main (void) {

        scanf ( "%d", &N );
        int x;
        while ( N -- ) {
                scanf ( "%d", &x );
                ++ cnt[ x ];
                }

        bool fl = 0;
        for ( int i=0; i<100005; ++i ) {
                if ( cnt[i] == 1 ) {
                        printf ( "%d%c", i, ( fl ? '\n' : ' ' ) );
                        fl = 1;
                        }
                }

}
```

Solution for 40 pts with sorting and binary search for n <= 200000

```
#include <cstdio>
#include <algorithm>

using namespace std;

typedef long long ll;

int main()
{
   int n;
   scanf ("%d",&n);
   if ( n <= 200000 )
```

```
{
   int m = (n>>1),a[100006],i,x,w=0,l,r,end,found;
   for ( i=1; i<=m; i++ ) scanf ("%d",&a[i]);
   sort (a+1,a+m+1); a[m+1] = -1;
   i=1;
   while ( i<=m )
   {
      if ( a[i] == a[i+1] ) i+=2;
      else
      {
         a[w+1] = a[i];
         w++;
         i++;
      }
   }
   end = w;
   for ( i=1; i<=m; i++ )
   {
      scanf ("%d",&x);



      l=1; r=w; found=-1;
      while ( l<=r )
      {
         if ( a[ (l+r)>>1 ] == x )
         {
            found = ((l+r)>>1);
            l = r+1;
         }
         else if ( a[ (l+r)>>1 ] < x ) l = ((l+r)>>1) + 1;
         else r = ((l+r)>>1) - 1;
      }

      if ( found == -1 )
      {
         end++; a[end] = x;
      }
      else a[found] = -1;
   }

   sort (a+1,a+end+1);

   i = 1; w = 0; a[end+1] = -1;
   while ( a[i] == -1 ) i++;
   while ( i<=end )
   {
      if ( a[i] == a[i+1] ) i+=2;
      else
      {
            a[w+1] = a[i];
            w++;
            i++;
          }
       }

       if ( a[1] < a[2] ) printf ("%d %d\n",a[1],a[2]);
       else printf ("%d %d\n",a[2],a[1]);
   }
}
```

Proposed solution:
We xor of all numbers (and result is X), with complexity O (N). Obviously X is equal to the xor of unique numbers. Why?
We know that X is not equal to 0, because there are two unique numbers . Let (X & (1 << j))! = 0, i.e., the bit position j is equal to 1.
Let us scan all the numbers and xor with X only numbers wit bit j also equal to 1. At the end of the operation we have received result Y, which will be equal to one of the two numbers. The two numbers that we are looking for are Y and X ^ Y.

```
#include <stdio.h>
long N;
long long M=0,K[63]={0};
int main()
{
   long long a=1,b,mask[63]={0};
   long i,j;

     for (i=0;i<63;i++){mask[i]=a;a<<=1;}
    scanf("%ld",&N);


 for (i=0;i<N;i++)
 {scanf("%I64d",&a);
   M^=a;
   for (j=0;j<63 && a>=mask[j];j++)if (a & mask[j]) K[j]^=a;
 }


for (i=0;i<63;i++) if (M & mask[i]) break;

 a=K[i];
 b=M^a;

if (a>b) {a=M;b=a^b;a=a^b;}
 printf("%I64d %I64d\n",a,b);

return 0;
}
```

Task 4. PARKING LOT

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

const int TYPE_BEGIN = 1;
const int TYPE_END = 2;

struct event {
	int time;
	int event_type;
};

bool operator<(event e1, event e2) {
	if(e1.time != e2.time)
		return e1.time < e2.time;
	return e2.event_type < e1.event_type;
}

int main()
{
	std::ios::sync_with_stdio(false);
	int n;
	cin >> n;
	vector<event> timeline;
	for(int i=0; i<n; i++) {
		int a, b;
		cin >> a >> b;
		event begin, end;
		begin.time = a;
		begin.event_type = TYPE_BEGIN;
		timeline.push_back(begin);
		end.time = b;
		end.event_type = TYPE_END;
		timeline.push_back(end);
	}
	sort(timeline.begin(), timeline.end());
	int occupied = 0;
	int capacity = 0;
	int amount = 0;
	int start = 0;
	for(event e: timeline) {
		if(e.event_type == TYPE_BEGIN) {
			occupied ++;
			if(occupied > capacity) {
				capacity = occupied;
```

```
                    amount = 0;
                }
                if(occupied == capacity) {
                    start = e.time;
                }
            }
            else { // e.event_type == TYPE_END
                if(occupied == capacity) {
                    amount += e.time - start;
                }
                occupied --;
            }
        }

        cout << amount;

        return 0;
}
```