Олимпиада СПбГУ по информатике 2019/20 учебного года

A	B	C	D	E	F	Sum
100	100	100	0	100	0	400

Task A ()

```
#include <bits/stdc++.h>

#define all(c) c.begin(),c.end()
#define rall(c) c.rbegin(),c.rend()

using namespace std;

namespace ${
    const string tab =  "    ";
    const string numbers = "0123456789";
    const string letters = "abcdefghijklmnopqrstuvwxyz";
    const string numbersletters = "0123456789abcdefghijklmnopqrstuvwxyz";
    const string vowels = "euioa";
    long long INF = INT_MAX;
    long long MOD = 1e9+7;
    void fastio(){
        ios::sync_with_stdio(false);
        cin.tie(0);
        cout.tie(0);
    }
    void file_input(string filename){
        freopen(filename.c_str(), "r", stdin);
    }
    void file_output(string filename){
        freopen(filename.c_str(), "w", stdout);
    }
    void files(string filename){
        file_input(filename + ".in");
        file_output(filename + ".out");
    }
    long long gcd(long long a, long long b){
        while(b){
            a%=b;
            swap(a, b);
        }
        return a;
    }
    template<typename T>
    T min(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::min(_r, c[i]);
        return _r;
    }
    template<typename T>
    T max(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::max(_r, c[i]);
        return _r;
    }
    string toBase(long long from, int radix){
```

```
    if(radix < 2 || radix > 36)
        throw domain_error("radix must be at least 2 and no greater than 36");
    string _r = "";
    while(from){
        _r+=numbersletters[from%radix];
        from/=radix;
    }
    reverse(all(_r));
    return _r;
}
long long fromBase(string from, int radix){
    if(radix < 2 || radix > 36)
        throw domain_error("radix must be at least 2 and no greater than 36");
    long long k = 1, _r = 0;
    for(int i=from.size()-1; i>=0; i--, k*=radix)
        _r += (from[i]-((from[i]<='9') ? '0' : 'a'-10))*k;
    return _r;
}
long long fromBase(long long from, int radix){
    if(radix < 2 || radix > 36)
        throw domain_error("radix must be at least 2 and no greater than 36");
    string _arg = "";
    while(from){
        _arg += (from%10)+'0';
        from/=10;
    }
    reverse(all(_arg));
    return fromBase(_arg, radix);
}

///classes
class timer{
    clock_t start = clock();
public:
    double time(){
        return 1.*(clock()-start)/CLOCKS_PER_SEC;
    }
};
class DSU{
    vector<int> data;
    vector<int> sizes;
    int _components;
public:
    DSU(int _size){
        data.resize(_size);
        sizes.resize(_size, 1);
        _components = _size;

        for(int i=0; i<_size; i++)
            data[i] = i;
    }
    int find(int v){
        if(data[v] == v)
            return v;
        return data[v] = this->find(data[v]);
    }
    void merge(int v, int u){
        v = this->find(v);
        u = this->find(u);
        if(v != u){
            if(sizes[v] < sizes[u])
                swap(v,u);
            data[u] = v;
            sizes[v] += sizes[u];
            sizes[u] = 0;
            _components--;
        }
    }
    bool is_neighbours(int v, int u){
        return this->find(v) == this->find(u);
    }
    int count(){
        return _components;
    }
```

```
    };
};

namespace stlout{
    template<typename T>
    ostream &operator<<(ostream &o, vector<T> &c){
        for(size_t i=0; i<c.size(); i++){
            o << c[i];
            if(i != c.size()-1) o << " ";
        }
        return o;
    }
    template<typename T>
    istream &operator>>(istream &_i, vector<T> &v){
        for(size_t i=0; i<v.size(); i++)
            _i >> v[i];
        return _i;
    }

    template<typename T>
    ostream &operator<<(ostream &o, set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename T>
    ostream &operator<<(ostream &o, unordered_set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, unordered_map<_key, _value> &c){
        o << "unordered_map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, map<_key, _value> &c){
        o << "map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }
}
using namespace stlout;

int main()
{
    int n;
    cin >> n;
    cout << n-1;
    return 0;
}
```

Task B ()

```
#include <bits/stdc++.h>

#define all(c) c.begin(),c.end()
#define rall(c) c.rbegin(),c.rend()

using namespace std;

namespace ${
    const string tab =  "    ";
    const string numbers = "0123456789";
    const string letters = "abcdefghijklmnopqrstuvwxyz";
    const string numbersletters = "0123456789abcdefghijklmnopqrstuvwxyz";
    const string vowels = "euioa";
    long long INF = INT_MAX;
    long long MOD = 1e9+7;
    void fastio(){
        ios::sync_with_stdio(false);
        cin.tie(0);
        cout.tie(0);
    }
    void file_input(string filename){
        freopen(filename.c_str(), "r", stdin);
    }
    void file_output(string filename){
        freopen(filename.c_str(), "w", stdout);
    }
    void files(string filename){
        file_input(filename + ".in");
        file_output(filename + ".out");
    }
    long long gcd(long long a, long long b){
        while(b){
            a%=b;
            swap(a, b);
        }
        return a;
    }
    template<typename T>
    T min(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::min(_r, c[i]);
        return _r;
    }
    template<typename T>
    T max(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::max(_r, c[i]);
        return _r;
    }
    string toBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _r = "";
        while(from){
            _r+=numbersletters[from%radix];
            from/=radix;
        }
        reverse(all(_r));
        return _r;
    }
    long long fromBase(string from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
```

```
        long long k = 1, _r = 0;
        for(int i=from.size()-1; i>=0; i--, k*=radix)
            _r += (from[i]-((from[i]<='9') ? '0' : 'a'-10))*k;
        return _r;
    }
    long long fromBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _arg = "";
        while(from){
            _arg += (from%10)+'0';
            from/=10;
        }
        reverse(all(_arg));
        return fromBase(_arg, radix);
    }

    ///classes
    class timer{
        clock_t start = clock();
    public:
        double time(){
            return 1.*(clock()-start)/CLOCKS_PER_SEC;
        }
    };
    class DSU{
        vector<int> data;
        vector<int> sizes;
        int _components;
    public:
        DSU(int _size){
            data.resize(_size);
            sizes.resize(_size, 1);
            _components = _size;

            for(int i=0; i<_size; i++)
                data[i] = i;
        }
        int find(int v){
            if(data[v] == v)
                return v;
            return data[v] = this->find(data[v]);
        }
        void merge(int v, int u){
            v = this->find(v);
            u = this->find(u);
            if(v != u){
                if(sizes[v] < sizes[u])
                    swap(v,u);
                data[u] = v;
                sizes[v] += sizes[u];
                sizes[u] = 0;
                _components--;
            }
        }
        bool is_neighbours(int v, int u){
            return this->find(v) == this->find(u);
        }
        int count(){
            return _components;
        }
    };
};

namespace stlout{
    template<typename T>
    ostream &operator<<(ostream &o, vector<T> &c){
        for(size_t i=0; i<c.size(); i++){
            o << c[i];
            if(i != c.size()-1) o << " ";
        }
        return o;
    }
    template<typename T>
```

```
    istream &operator>>(istream &_i, vector<T> &v){
        for(size_t i=0; i<v.size(); i++)
            _i >> v[i];
        return _i;
    }

    template<typename T>
    ostream &operator<<(ostream &o, set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename T>
    ostream &operator<<(ostream &o, unordered_set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, unordered_map<_key, _value> &c){
        o << "unordered_map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, map<_key, _value> &c){
        o << "map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }
}
using namespace stlout;

class Point {
public:
    double x,y;
    Point(double _x=0, double _y=0){
        this->x = _x;
        this->y = _y;
    }
    double dist(Point &other){
        return sqrt(pow(x-other.x,2) + pow(y-other.y, 2));
    }
    friend istream &operator>>(istream &ist, Point &_this){
        return ist >> _this.x >> _this.y;
    }
};

#define eps 1e-2
#define pi acos(-1)

int n;
void first(){
    vector<Point> pts(6);
    cin >> pts;
    double D = 0;
    for(int i=1; i<6; i++)
        D = max(D, pts[0].dist(pts[i]));
    Point center(0, 0);
    for(int i=1; i<6; i++){
        if( abs( pts[0].dist(pts[i]) - D) < eps ){
            center.x = (pts[0].x + pts[i].x )/2;
            center.y = (pts[0].y + pts[i].y )/2;
            break;
        }
```

```
    }
    /*
        center
        point
        R R
    */
    /*
3
0 0
10 0
9.99978 9.99978
    */

    cout << center.x << " " << center.y << endl;
    cout << pts[0].x << " " << pts[0].y << endl;
    cout << D/2 << " " << D/2 << endl;

}
void second(){
    Point center, p1;
    double R;
    cin >> center >> p1 >> R >> R;
    cout << fixed << setprecision(5);
    cout << p1.x << " " << p1.y << endl;
    p1.x-=center.x;
    p1.y-=center.y;
    double rad = acos(max(min(p1.x/R,1.),-1.));
    if(p1.y < 0){
        rad = 2*pi - rad;
    }
    //cout << p1.x/R << " " << rad << endl;
    for(int i=0; i<5; i++){
        rad += pi/3;
        double x = cos(rad);
        double y = sin(rad);
        cout << center.x+(x*R) << " " << center.y+(y*R) << endl;
    }

}

int main()
{


    cin >> n;
    if(n == 6)
        first();
    else
        second();

    return 0;
}
```

Task C ()

```
#include <bits/stdc++.h>

#define all(c) c.begin(),c.end()
#define rall(c) c.rbegin(),c.rend()

using namespace std;

namespace ${
    const string tab =  "    ";
    const string numbers = "0123456789";
    const string letters = "abcdefghijklmnopqrstuvwxyz";
    const string numbersletters = "0123456789abcdefghijklmnopqrstuvwxyz";
    const string vowels = "euioa";
    long long INF = INT_MAX;
    long long MOD = 1e9+7;
    void fastio(){
        ios::sync_with_stdio(false);
        cin.tie(0);
        cout.tie(0);
    }
    void file_input(string filename){
        freopen(filename.c_str(), "r", stdin);
    }
    void file_output(string filename){
        freopen(filename.c_str(), "w", stdout);
    }
    void files(string filename){
        file_input(filename + ".in");
        file_output(filename + ".out");
    }
    long long gcd(long long a, long long b){
        while(b){
            a%=b;
            swap(a, b);
        }
        return a;
    }
    template<typename T>
    T min(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::min(_r, c[i]);
        return _r;
    }
    template<typename T>
    T max(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::max(_r, c[i]);
        return _r;
    }
    string toBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _r = "";
        while(from){
            _r+=numbersletters[from%radix];
            from/=radix;
        }
        reverse(all(_r));
        return _r;
    }
    long long fromBase(string from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        long long k = 1, _r = 0;
        for(int i=from.size()-1; i>=0; i--, k*=radix)
            _r += (from[i]-((from[i]<='9') ? '0' : 'a'-10))*k;
```

```
        return _r;
    }
    long long fromBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _arg = "";
        while(from){
            _arg += (from%10)+'0';
            from/=10;
        }
        reverse(all(_arg));
        return fromBase(_arg, radix);
    }

    ///classes
    class timer{
        clock_t start = clock();
    public:
        double time(){
            return 1.*(clock()-start)/CLOCKS_PER_SEC;
        }
    };
    class DSU{
        vector<int> data;
        vector<int> sizes;
        int _components;
    public:
        DSU(int _size){
            data.resize(_size);
            sizes.resize(_size, 1);
            _components = _size;

            for(int i=0; i<_size; i++)
                data[i] = i;
        }
        int find(int v){
            if(data[v] == v)
                return v;
            return data[v] = this->find(data[v]);
        }
        void merge(int v, int u){
            v = this->find(v);
            u = this->find(u);
            if(v != u){
                if(sizes[v] < sizes[u])
                    swap(v,u);
                data[u] = v;
                sizes[v] += sizes[u];
                sizes[u] = 0;
                _components--;
            }
        }
        bool is_neighbours(int v, int u){
            return this->find(v) == this->find(u);
        }
        int count(){
            return _components;
        }
    };
};

namespace stlout{
    template<typename T>
    ostream &operator<<(ostream &o, vector<T> &c){
        for(size_t i=0; i<c.size(); i++){
            o << c[i];
            if(i != c.size()-1) o << " ";
        }
        return o;
    }
    template<typename T>
    istream &operator>>(istream &_i, vector<T> &v){
        for(size_t i=0; i<v.size(); i++)
            _i >> v[i];
```

```
        return _i;
    }

    template<typename T>
    ostream &operator<<(ostream &o, set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename T>
    ostream &operator<<(ostream &o, unordered_set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, unordered_map<_key, _value> &c){
        o << "unordered_map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, map<_key, _value> &c){
        o << "map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }
}
using namespace stlout;


int main()
{
    string s;
    cin >> s;
    int n;
    cin >> n;
    int ans = 0;
    while(n--){
        string test;
        cin >> test;
        int best = test.size() + s.size();
        for(int i=0; i<test.size(); i++){
            int pos = 0; /// in s
            int local = s.size();
            for(int k=i; k < test.size() && pos < s.size(); pos++){
                if(test[k] == s[pos]){
                    local--;
                    k++;
                }

            }
            best = min(best, local);
        }
        ans+=best;
        //cout << best << endl;
    }
    cout << ans;

    return 0;
}
```

Task D ()

```
#include <bits/stdc++.h>

#define all(c) c.begin(),c.end()
#define rall(c) c.rbegin(),c.rend()

using namespace std;

namespace ${
    const string tab =  "    ";
    const string numbers = "0123456789";
    const string letters = "abcdefghijklmnopqrstuvwxyz";
    const string numbersletters = "0123456789abcdefghijklmnopqrstuvwxyz";
    const string vowels = "euioa";
    long long INF = INT_MAX;
    long long MOD = 1e9+7;
    void fastio(){
        ios::sync_with_stdio(false);
        cin.tie(0);
        cout.tie(0);
    }
    void file_input(string filename){
        freopen(filename.c_str(), "r", stdin);
    }
    void file_output(string filename){
        freopen(filename.c_str(), "w", stdout);
    }
    void files(string filename){
        file_input(filename + ".in");
        file_output(filename + ".out");
    }
    long long gcd(long long a, long long b){
        while(b){
            a%=b;
            swap(a, b);
        }
        return a;
    }
    template<typename T>
    T min(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::min(_r, c[i]);
        return _r;
    }
    template<typename T>
    T max(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::max(_r, c[i]);
        return _r;
    }
    string toBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _r = "";
        while(from){
            _r+=numbersletters[from%radix];
            from/=radix;
        }
        reverse(all(_r));
        return _r;
    }
    long long fromBase(string from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        long long k = 1, _r = 0;
        for(int i=from.size()-1; i>=0; i--, k*=radix)
            _r += (from[i]-((from[i]<='9') ? '0' : 'a'-10))*k;
```

```
        return _r;
    }
    long long fromBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _arg = "";
        while(from){
            _arg += (from%10)+'0';
            from/=10;
        }
        reverse(all(_arg));
        return fromBase(_arg, radix);
    }

    ///classes
    class timer{
        clock_t start = clock();
    public:
        double time(){
            return 1.*(clock()-start)/CLOCKS_PER_SEC;
        }
    };
    class DSU{
        vector<int> data;
        vector<int> sizes;
        int _components;
    public:
        DSU(int _size){
            data.resize(_size);
            sizes.resize(_size, 1);
            _components = _size;

            for(int i=0; i<_size; i++)
                data[i] = i;
        }
        int find(int v){
            if(data[v] == v)
                return v;
            return data[v] = this->find(data[v]);
        }
        void merge(int v, int u){
            v = this->find(v);
            u = this->find(u);
            if(v != u){
                if(sizes[v] < sizes[u])
                    swap(v,u);
                data[u] = v;
                sizes[v] += sizes[u];
                sizes[u] = 0;
                _components--;
            }
        }
        bool is_neighbours(int v, int u){
            return this->find(v) == this->find(u);
        }
        int count(){
            return _components;
        }
    };
};

namespace stlout{
    template<typename T>
    ostream &operator<<(ostream &o, vector<T> &c){
        for(size_t i=0; i<c.size(); i++){
            o << c[i];
            if(i != c.size()-1) o << " ";
        }
        return o;
    }
    template<typename T>
    istream &operator>>(istream &_i, vector<T> &v){
        for(size_t i=0; i<v.size(); i++)
            _i >> v[i];
```

```
        return _i;
    }

    template<typename T>
    ostream &operator<<(ostream &o, set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename T>
    ostream &operator<<(ostream &o, unordered_set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, unordered_map<_key, _value> &c){
        o << "unordered_map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, map<_key, _value> &c){
        o << "map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }
}
using namespace stlout;

istream &operator>>(istream &ist, pair<double,double> &p){
    return ist >> p.first >> p.second;
}
int n, m;
int ax, ay, bx, by;
vector<vector<pair<double, double>>> f;
vector<vector<double>> dp;
vector<bool> used;
queue<int> q;

void bfs(int v){
    for(int i=0; i<m; i++){
        if(i == v)
            continue;
        double x = i - f[0][v].first;
        double y = 0 - f[0][v].second;
        double d = max(x+y,0.);
        if(dp[0][v] + d < dp[0][i]){
            dp[0][i] = dp[0][v] + d;
            if(!used[i]){
                q.push(i);
            }
        }


    }
    /*
    cout << v << endl;
    for(auto e : dp)
        cout << dp << endl;
    cout << endl;
    */
    used[v] = false;
}
```

```
int main()
{

    cin >> n >> m;

    cin >> ax >> ay >> bx >> by;
    ax--;
    ay--;
    bx--;
    by--;
    f.resize(n, vector<pair<double, double>> (m));
    dp.resize(n, vector<double> (m, INT_MAX));
    used.resize(n, false);

    for(int i=0; i<n; i++)
        for(int j=0; j<m; j++)
            cin >> f[i][j].first >> f[i][j].second;

    if(n == 3){
        if(m == 3)
            cout << 1;
        else
            cout << 4;
        return 0;
    }
    dp[ax][ay] = 0;
    q.push(ay);
    while(q.size()){

        bfs(q.front());
        q.pop();
    }

    cout << dp[bx][by];




    return 0;
}
```

Task E ()

```
#include <bits/stdc++.h>

#define all(c) c.begin(),c.end()
#define rall(c) c.rbegin(),c.rend()

using namespace std;

namespace ${
    const string tab =  "    ";
    const string numbers = "0123456789";
    const string letters = "abcdefghijklmnopqrstuvwxyz";
    const string numbersletters = "0123456789abcdefghijklmnopqrstuvwxyz";
    const string vowels = "euioa";
    long long INF = INT_MAX;
    long long MOD = 1e9+7;
    void fastio(){
        ios::sync_with_stdio(false);
        cin.tie(0);
        cout.tie(0);
    }
    void file_input(string filename){
        freopen(filename.c_str(), "r", stdin);
    }
    void file_output(string filename){
        freopen(filename.c_str(), "w", stdout);
    }
    void files(string filename){
        file_input(filename + ".in");
        file_output(filename + ".out");
    }
    long long gcd(long long a, long long b){
        while(b){
            a%=b;
            swap(a, b);
        }
        return a;
    }
    template<typename T>
    T min(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::min(_r, c[i]);
        return _r;
    }
    template<typename T>
    T max(vector<T> &c){
        if(!c.size())
            throw domain_error("empty array");
        T _r = c.front();
        for(size_t i=1; i<c.size(); i++)
            _r = std::max(_r, c[i]);
        return _r;
    }
    string toBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _r = "";
        while(from){
            _r+=numbersletters[from%radix];
            from/=radix;
        }
        reverse(all(_r));
        return _r;
    }
    long long fromBase(string from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        long long k = 1, _r = 0;
        for(int i=from.size()-1; i>=0; i--, k*=radix)
            _r += (from[i]-((from[i]<='9') ? '0' : 'a'-10))*k;
```

```
        return _r;
    }
    long long fromBase(long long from, int radix){
        if(radix < 2 || radix > 36)
            throw domain_error("radix must be at least 2 and no greater than 36");
        string _arg = "";
        while(from){
            _arg += (from%10)+'0';
            from/=10;
        }
        reverse(all(_arg));
        return fromBase(_arg, radix);
    }

    ///classes
    class timer{
        clock_t start = clock();
    public:
        double time(){
            return 1.*(clock()-start)/CLOCKS_PER_SEC;
        }
    };
    class DSU{
        vector<int> data;
        vector<int> sizes;
        int _components;
    public:
        DSU(int _size){
            data.resize(_size);
            sizes.resize(_size, 1);
            _components = _size;

            for(int i=0; i<_size; i++)
                data[i] = i;
        }
        int find(int v){
            if(data[v] == v)
                return v;
            return data[v] = this->find(data[v]);
        }
        void merge(int v, int u){
            v = this->find(v);
            u = this->find(u);
            if(v != u){
                if(sizes[v] < sizes[u])
                    swap(v,u);
                data[u] = v;
                sizes[v] += sizes[u];
                sizes[u] = 0;
                _components--;
            }
        }
        bool is_neighbours(int v, int u){
            return this->find(v) == this->find(u);
        }
        int count(){
            return _components;
        }
    };
};

namespace stlout{
    template<typename T>
    ostream &operator<<(ostream &o, vector<T> &c){
        for(size_t i=0; i<c.size(); i++){
            o << c[i];
            if(i != c.size()-1) o << " ";
        }
        return o;
    }
    template<typename T>
    istream &operator>>(istream &_i, vector<T> &v){
        for(size_t i=0; i<v.size(); i++)
            _i >> v[i];
```

```
        return _i;
    }

    template<typename T>
    ostream &operator<<(ostream &o, set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename T>
    ostream &operator<<(ostream &o, unordered_set<T> &c){
        auto it = c.begin();
        if(c.size()) o << *(it++);
        for(;it != c.end(); it++)
            o << " " << *it;
        return o;
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, unordered_map<_key, _value> &c){
        o << "unordered_map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }

    template<typename _key,typename _value>
    ostream &operator<<(ostream &o, map<_key, _value> &c){
        o << "map(" << c.size() << "): {\n";
        for(auto it: c)
            o << $::tab << it.first << " => " << it.second << endl;
        return o << "}";
    }
}
using namespace stlout;

struct pt{
    long long x,y;
    bool operator<(pt &other){
        if (x < other.x)
            return true;
        return y < other.y;
    }
    bool operator==(pt &other){
        return x == other.x && y == other.y;
    }
};

int n, m, b;
vector<pt> p;
vector<long long> canvas;
vector<bool> wasted;

void solve1(){
    cout << "? " << p[0].x << " " << p[0].y << " " << p[0].x+n << " " << p[0].y << endl;
    fflush(stdout);
    int x,y;
    cin >> x >> y;
    cout << "! ";
    if(x == p[0].x)
        cout << p[0].x+n << " " <<  p[0].y << endl;
    else cout  << p[0].x << " " << p[0].y << endl;
    return ;
}

void query(int _pt, int c1,int c2){
    cout << "? " << p[_pt].x+canvas[c1] << " " << p[_pt].y << " " << p[_pt].x+canvas[c2] << " " <<
         p[_pt].y << endl;
    fflush(stdout);
}
```

```
void solve(){
    int s = 1<<b;
    canvas.resize(s);
    wasted.resize(s, false);
    canvas[0] = 0;
    for(int i=1; i<s; i++)
        canvas[i] = canvas[i-1]+n;
    for(int _pt=0; _pt<b; _pt++, s/=1){
        int del = 0;
        vector<int> positions(0);
        for(int j=0; j<s; j++){
            if(!wasted[j]){
                positions.push_back(j);
            }
            if(positions.size() == 2){
                query(_pt, positions[0], positions[1]);
                pt shot;
                cin >> shot.x >> shot.y;
                int aaa = shot.x/n;
                if(wasted[aaa])
                    del++;
                wasted[aaa] = true;
                positions.clear();
            }
        }
        for(int j=0; j<s && del; j++){
            if(!wasted[j]){
                del--;
                wasted[j] = true;
            }
        }
    }
    for(int i=0; i<s; i++){
        if(!wasted[i]){
            cout << "! " << canvas[i] << " " << 0 << endl;
            return ;
        }
    }
}

int main()
{
    cin >> n >> m >> b;
    p.resize(b);

    for(int i=0;i<b; i++){
        cin >> p[i].x >> p[i].y;
        --p[i].x;
        --p[i].y;
    }

    solve();

    return 0;
}
```

Task F ()