A. Ф. И. О.

B. Шахматы

Заметим, что полное решение включает в себя вычисление по формуле. Самое сложное – вычислить число клеток на диагонали. Необходимо помнить, что на диагоналях для всех клеток диагоналей либо сумма, либо разность координат одинаковая (такая же, как у клетки, на которой находится ферзь). Приведем решение на языке C++.

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

using ll = long long;

using namespace std;

int main(){

ll x , y, m, n;

ll ans = 0;

cin >> m >> n >> x >> y;

ans += n - 1;

ans += m - 1;

ans += min(n, m - x + y) - max(1ll, 1 - x + y) + 1;

ans +=  min(n, x + y - 1) - max(1ll, x + y - m) + 1;

cout << ans - 2;

}
```

C. Последовательности

Приведем полное эффективное решение на языке C++, в котором реализуется идея динамического программирования, а именно подсчитываются последовательности, для которых количество нулей равно i, а количество единиц = j, то есть параметрами динамики будут количество нулей и единиц.

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

using LL = long long;

using namespace std;

LL w[30][30];

int n, K;

LL ans;

LL cnt(int i, int j){

   if (i + j  == 0){
```

```
        w[i][j] = 1;
        return 1;
    }
    if (w[i][j] > 0)
        return w[i][j];
    if (i > 0) {
    w[i][j] += cnt(i - 1, j);
    }
    if (j > 0){
      w[i][j] += cnt(i, j - 1);
    }
    return w[i][j];
}
int main(){
    cin >> n >> K;
    for (int i = 0; i <= min(K, n); i++)
        for (int j = 0; j <= min(K, n); j++)
          if (i + j> 0 && i + j == n){
              ans += cnt(i, j);
          }
    cout << ans;
}


cout << ans;
}
```

D. Проверка памяти

Приведем полное решение на языке Python. Эффективное решение включает в себя использование структуры set. Кроме того, поскольку все числа не превосходят 10000, достаточно проверять делимость в цикле от 1 до квадратного корня из 10000, который равен 100.

```
n = int(input())
se = set()
res = 0
a = list(map(int, input().split()))
```

```
for i in range(n):
    x = a[i]
    for j in range(1, 101):
        if (x % j == 0) and (j in se  or x// j in se) :
            res += 1
            break
    se.add(x)
print(res)
```

E. Наоборот

Приведем полное решение на языке C++. Эффективное решение основано на методе сортировки подсчетом. Необходимо делать проверку строк на то, что они красивые, и пересчитывать в цикле число красивых строк, которые заканчиваются на каждую из букв латинского алфавита, чтобы потом вычислять, сколько красивых строк могут быть склеены с последующими красивыми строками.

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
int n;
ll ans = 0;
int st[26], f[26];

bool isGood(const string& s){
 char ol = 'a';
 for (auto c: s){
    if (ol > c)
        return false;
    ol = c;
 }
 return true;
}
int main(){
    cin >> n;
    string s;
    for (int i = 0; i < n; i++){
```

```
    cin >> s;
    char en = s.back();
    if (isGood(s))
    {
        for (int i = 0; i <= s[0] -'a' ; i++)
            ans += f[i];
        f[en -'a']++;
    }
  }
```

F. Постаматы

Приведем полное решение на языке C++. Эффективное решение включает в себя использование структуры set.

```
#include <iostream>
#include <set>
using namespace std;
typedef long long ll;
int main(){
 ios::sync_with_stdio(0); // для ускорения ввода
 cin.tie(0);
 int n, a, q, k, x;
 cin >> n;
 set<pair<int, int>> se;
 for(int i = 1; i <= n; ++i){
    int a;
    cin >> a;
    se.insert({i, a});
 }
 cin >> q;

 for(int i = 0; i < q; ++i){
    ll d = 0;
```

```
    cin >> x >> k;
    auto it = se.lower_bound({x, 0});
    if(it == se.end()) {
       it = se.begin();
       x -= n;
     }
    while(k > 0){
       int f = it->first;
       int s = it->second;
       auto itr = it;
       it++;
       se.erase(itr);
       if(s > k){
          d += (ll)k*(f-x);
          se.insert({f, s - k});

       }
       else
       {
          d += (ll)s*(f-x);
          if(it == se.end()) {
             it = se.begin();
             x -= n;

          }
       }
       k -= s;
    }
 cout << d << "\n";
 }
}
```