Контрольная работа. Функции.

Во всех задачах приведена некоторая программа, в коде которой используется одна или несколько функций. Эта программа *правильно* решает поставленную задачу при условии, что функции, ей используемые, реализованы корректно.
Вам надо сдать в тестирующую систему файл, в котором определены эти функции так, чтобы приведённая программа верно решала поставленную задачу.

A. Приведена программа, считывающая два натуральных числа A и B и печатающая несократимую дробь, равную $\frac{A}{B}$.

```
#include<iostream>
```

```
//... your code here
```

```
int main() {
    long long x, y, g;
    std::cin >> x >> y;
    g = gcd(x, y);
    std::cout << x / g << '/' << y / g << "\n";
}
```

Input	Output
345 3672456	5/53224

B. Известен следующий математический факт: несократимая обыкновенная дробь представляется в виде конечной десятичной дроби в том и только в том случае, когда её знаменатель в разложении на простые множители имеет только числа 2 и 5.

Приведена программа, которая выводит `YES` в случае, если дробь $\frac{N}{K}$ представляется в виде конечной десятичной дроби и выводит `NO` в противном случае.

```
#include<iostream>
```

```
//... your code here
```

```
int main() {
    long long n, k;
    std::cin >> n >> k;
    long long g = gcd(n, k);
    if (divisibility_2_5(k / g)) {
        std::cout << "YES";
    } else {
        std::cout << "NO";
    }
}
```

Input	Output
3 6	YES
3 7	NO
13 160	YES

C. Приведена программа, относительно эффективно вычисляющая биномиальный коэффициент, используя формулу:

$$C_N^K = \frac{N!}{(N-K)!K!}$$

```
#include<iostream>
```

```
//... your code here
```

```
int main() {
    long long N, K, d1, d2, d3, cnk = 1;
    std::cin >> N >> K;
    for (int p = 2; p <= N; p++) {
        if (is_prime(p)) {
            d1 = prime_multiplicity(N, p);
            d2 = prime_multiplicity(N - K, p);
            d3 = prime_multiplicity(K, p);
            cnk *= power(p, d1 - d2 - d3);
        }
    }
    std::cout << cnk << "\n";
}
```

Input	Output
4 2	6
16 10	8008

D. Приведена программа, которая вычисляет арифметическое выражение, введённое в виде символьной строки. Выражение содержит только целые неотрицательные числа и знаки сложения и вычитания.
Числа не превосходят $2 \cdot 10^6$, их количество не превосходит 10^5.

```
#include<iostream>
#include<string>
```

```
//... your code here
```

```
int main() {
    std::string s;
    std::cin >> s;
    current_number cur_num;
    int sign = 1;
    if (s[0] == '-') {
        sign = -1;
    }
    cur_num = shift(s, 0);
    long long ans = cur_num.number;
    while (cur_num.pos < s.size()) {
        sign = s[cur_num.pos];
        cur_num.pos++;
        cur_num = shift(s, cur_num.pos);
        if (sign == '+') {
            ans += cur_num.number;
        } else {
            ans -= cur_num.number;
        }
    }
    std::cout << ans << "\n";
}
```

Input	Output
1+12-54+68-17	10
-17-17	-34

E. Ниже приведён один из возможных *наивных* алгоритмов умножения многочленов. Каждый многочлен задан массивом коэфициентов, где $x[k]$ — коэффициент при одночлене x^k.
В первой строке вводятся два числа: степени многочленов.
Во второй строке вводятся коэффициенты первого многочлена, в третьей строке коэффициенты второго многочлена. Коэффициенты вводятся, начиная от свободного члена, т.е. коэффициента при x^0.

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

```
//... your code here
```

```
int main() {
    int N, K;
    std::cin >> N >> K;
    std::vector<long long> x(N + 1, 0), y(K + 1, 0);
    for (int i = 0; i < N + 1; i++) {
        std::cin >> x[i];
    }
    for (int i = 0; i < K + 1; i++) {
        std::cin >> y[i];
    }
    std::vector<long long> res(x.size() + y.size() - 1);

    for (int k = 0; k < y.size(); ++k) {
        std::vector<long long> tmp = x;
        shift(tmp, k);
        mult_const(tmp, y[k]);
        add(res, tmp);
    }
    print(res);
}
```

Input	Output
0 0 2 3	6
1 1 -2 1 3 1	-6 1 1
2 3 -3 0 1 -7 2 0 -1	21 -6 -7 5 0 -1

Комментарий ко второму тесту: $(x-2)(x+3) = x^2 + x - 6$