Заключительный этап

Индивидуальный предметный тур

Математика. 8-9 класс

Задача III.1.1.1. (15 баллов)

Вася выписал все четные натуральные числа от 2 до 10 000 по одному разу 24681012...999810000. Сколько раз он записал цифру 5?

Решение

Поскольку цифра 5 — нечетная, она не может быть последней цифрой четного числа.

1. Для чисел от 2 до 98 цифра 5 может встретиться только как первая цифра двузначного числа, т. е. пять раз — в числах 50, 52, 54, 56 и 58.
2. Аналогично — для чисел от 100 до 198, от 200 до 298, от 300 до 398, от 400 до 498, от 600 до 698, от 700 до 798, от 800 до 898 и от 900 до 998. В каждом из указанных восьми наборов по 50 чисел цифра 5 встретится ровно пять раз, итого дополнительно 40 раз.
3. Для чисел от 500 до 598 в дополнение к пяти случаям, встречающимся в каждом наборе из 50 чисел, цифра 5 используется как первая цифра дополнительно в каждом числе — итого $50 + 5 = 55$ раз.
4. Итого во всех четных числах от 2 до 998 цифра 5 используется $5+40+55 = 100$ раз.
5. В числе 1000 цифра 5 не используется. В числах от 1000 до 1998 цифра 5 встречается столько же раз, сколько в наборе чисел от 2 до 998, т. е. 100 раз. Аналогично — в наборах от 1000 до 1998, от 2000 до 2998, от 3000 до 3998, от 4000 до 4998, от 6000 до 6998, от 7000 до 7998, от 8000 до 8998 и от 9000 до 9998. Итого в дополнение к п. 4 получаем еще 800 раз, всего 900. В числе 10 000 цифра 5 не используется.
6. Наконец, четные числа от 5002 до 6000 дают, помимо 100 «стандартных» раз, дополнительно еще 500 раз цифру 5 (первая цифра числа). Т. е. еще 600 раз.
7. Итого: $900 + 600 = 1500$.

Система оценки

- Полное обоснованное решение — 15 баллов.
- Имеются частичные продвижения, но итоговое количество посчитано неверно — 5 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: 1500.

Задача III.1.1.2. (15 баллов)

Петя выбрал положительное целое число n. Затем он выбрал натуральный делитель числа n, умножил его на 5, вычел этот результат из n и получил 2021. Найдите все числа, которые мог выбрать Петя. (Заметим, что 2021 — составное число.)

Решение

Разложим 2021 на простые множители: $2021 = 43 \cdot 47$. Пусть d — выбранный натуральный делитель числа n. Значит, $n = d \cdot k$. По условию $n - 5d = 2021$, $d \cdot (k - 5) = 2021$. Отсюда возможны варианты:

$$\begin{cases} d = 1 \\ k - 5 = 2021 \end{cases}$$

$$\begin{cases} d = 43 \\ k - 5 = 47 \end{cases}$$

$$\begin{cases} d = 47 \\ k - 5 = 43 \end{cases}$$

$$\begin{cases} d = 2021 \\ k - 5 = 1 \end{cases}$$

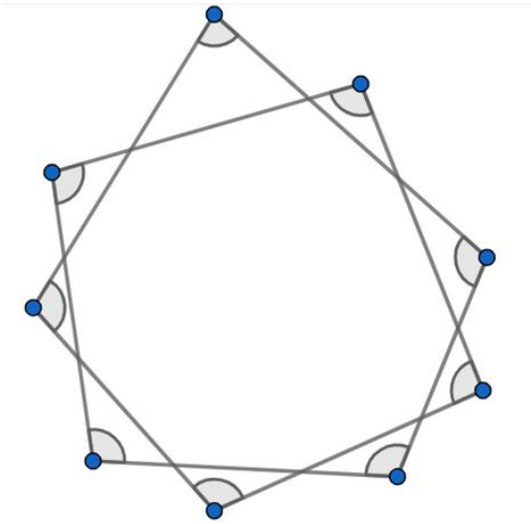
Система оценки

- Полное обоснованное решение — 15 баллов.
- Получено уравнение «$d \cdot (k - 5) = 2021$» — 5 баллов.
- Найдены некоторые (не все) значения n — 5 баллов. (В дополнение к пред. пункту).
- Задача не решена или решена неверно — 0 баллов.

Ответ: 2026; $43 \cdot 52$; $47 \cdot 48$; $2021 \cdot 6$.

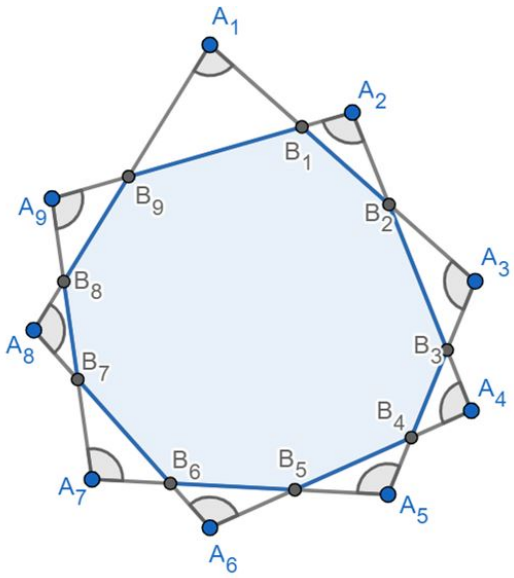
Задача III.1.1.3. (20 баллов)

Найдите сумму отмеченных углов 9-конечной звезды. (Отмеченные углы не обязательно равны; звезда произвольная.)



Решение

Обозначим вершины звезды $A_1, A_2, ..., A_9$, а точки пересечения звеньев $B_1, B_2, ..., B_9$ (см. рисунок).



Сумма внешних углов произвольного выпуклого многоугольника, взятых по одному разу при каждой вершине, равна 360°. Значит, сумма всех внешних углов выпуклого 9-угольника $B_1B_2...B_9$ взятых по два раза при каждой вершине, составляет 720°. Но эти углы вместе с углами $A_1, A_2, ..., A_9$ составляют внутренние углы девяти треугольников $A_1B_1B_9$, $A_2B_2B_1$,...,$A_9B_9B_8$. Значит, искомая сумма

$$\angle A_1 + \angle A_2 + ... + \angle A_9 = 180^\circ \cdot 9 - 720^\circ = 900^\circ.$$

Система оценки

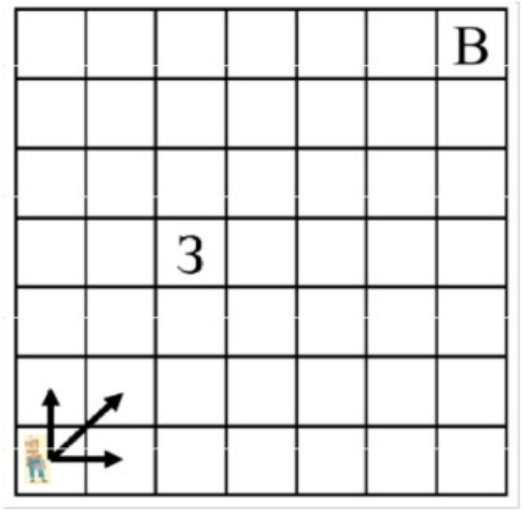
- Полное обоснованное решение — 20 баллов.
- Приведен верный ответ, при этом могут быть частичные продвижения — 10 баллов.

- При верном решении получен неверный ответ из-за вычислительной ошибки — 15 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: 900°.

Задача III.1.1.4. (20 баллов)

Парк, имеющий форму квадрата, разбит на 49 одинаковых квадратов. (См. чертеж.) Робот-садовник находится в левом нижнем квадратике парка. Он должен дойти по парку до верхнего правого квадратика (В), обязательно побывав в квадратике с заправкой (З). Робот может за один ход пойти вверх в соседний квадрат, вправо в соседний квадрат или вправо-вверх по диагонали в соседний (по вершине) квадрат. Сколько возможных маршрутов существует?



Решение

Обозначим шаг вверх буквой В, шаг вправо — П, шаг по диагонали — Д. Найдем количество маршрутов от начальной точки до квадратика (З).

1. Самый короткий путь состоит из двух диагональных ходов (2Д) и одного хода вверх (1В) в любом порядке. Это маршрут вида 2Д + 1В. Их всего 3.
2. Один диагональный ход Д можно заменить на один В и один П. Это маршрут вида Д + 2В + 1П. Их количество вычислим по формуле для перестановок с повторениями $\frac{(1+2+1)!}{1!\cdot 2!\cdot 1!} = 12$.
3. Еще один диагональный ход Д можно заменить на один В и один П. Это маршрут самый длинный. Он имеет вид 3В + 2П. Их количество равно $\frac{(3+2)!}{3!\cdot 2!} = 10$.

Всего маршрутов $3 + 12 + 10 = 25$.

Найдем количество маршрутов от квадратика (З) до конечного квадратика (В).

1. Самый короткий путь состоит из трех диагональных ходов (3Д) и одного хода вправо (1П) в любом порядке. Это маршрут вида 3Д + 1П. Их всего 4.
2. Один диагональный ход Д можно заменить на один В и один П. Это маршрут вида 2Д + В + 2П. Их количество $\frac{(2+2+1)!}{2!\cdot 2!\cdot 1!} = 30$.

3. Еще один диагональный ход Д можно заменить на один В и один П. Этот маршрут имеет вид 1Д + 2В + 3П. Их количество равно $\frac{(1+2+3)!}{1!\cdot 2!\cdot 3!} = 60$.
4. Еще один диагональный ход Д можно заменить на один В и один П. Это маршрут самый длинный. Он имеет вид 3В + 4П. Их количество равно $\frac{(3+4)!}{3!\cdot 4!} = 35$.

Всего маршрутов $4 + 30 + 60 + 35 = 129$.

Количество маршрутов от начальной точки до конечного квадрата (В) находим, перемножая $25 \cdot 129 = 3225$.

Система оценки

- Полное обоснованное решение — 20 баллов.
- Найдены возможные виды маршрутов и количество каких-либо типов вычислено — 5 баллов.
- Верно найдено количество маршрутов до точки (З), либо от (З) до (В) — 10 баллов.
- Найдены количества маршрутов до точки (З) и от (З) до (В), но количества сложены — 15 баллов.
- Задача решена с разбором всех случаев, но содержит вычислительную ошибку — 15 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: 3225.

Задача III.1.1.5. (30 баллов)

Найдите все натуральные числа вида $\overline{\underbrace{yy...y}_{n}\underbrace{zz...z}_{n}}$ (n раз подряд идет цифра y, затем n раз подряд цифра z), которые для какого-нибудь натурального $n > 1$ являются квадратом натурального числа.

Решение

Поскольку $n \geqslant 2$ то квадрат натурального числа заканчивается по крайней мере на две одинаковые цифры. Исходя из того, что квадрат может заканчиваться только на цифры 0; 1; 4; 5; 6; 9 и он дает остаток 0 или 1 при делении на 4, то эти две цифры могут быть только 0 или 4. Очевидно, что 0 не может быть, потому что тогда $\overline{yy...y}$ — точный квадрат. А это невозможно, что нетрудно проверить (например, по остаткам от деления на 3 и 4).

Таким образом, $z = 4$ Преобразуем $x^2 = \overline{\underbrace{yy...y}_{n}\underbrace{44...4}_{n}}$;

$$x^2 = \underbrace{11...1}_{n} \cdot (10^n y + 4) \Rightarrow (\frac{x}{2})^2 = \underbrace{11...1}_{n} \cdot (25 \cdot 10^{n-2} y + 1)$$

При $n \geqslant 4$ правая часть $(\frac{x}{2})^2 = \underbrace{11...1}_{n} \cdot (25 \cdot 10^{n-2} y + 1)$ дает остаток 3 при делении на 4, а левая часть является квадратом, что невозможно.

При $n = 3$ имеем квадратом число $111 \cdot (1000y+4)$, а это означает, что обязательно $1000y + 4 \vdots 111$, что невозможно ни для какой цифры y.

Поэтому $n = 2$. Аналогично $11 \cdot (100y + 4)$ является точным квадратом, а это означает, что обязательно $100y + 4 \vdots 11$, откуда $y = 7$. Получаем ответ: $7744 = 88^2$.

Система оценки

- Полное обоснованное решение — 30 баллов.
- Записан верный ответ, но решение отсутствует (не доказано, что решение единственное) — 5 баллов.
- Продвижение (баллы суммируются):
  - доказано, что или 4: 5 баллов;
  - доказано, что 5 баллов;
  - доказано, что 10 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: 7744.

Математика. 10-11 класс

Задача III.1.2.1. (15 баллов)

Найдите наименьшее положительное целое число, в котором, по крайней мере, две цифры, и которое после зачеркивания его первой (то есть самой левой) цифры число уменьшается в 57 раз.

Решение

Пусть искомое число $A = a \cdot 10^n$, где a — первая цифра, а b — n-значное число. По условию $a \cdot 10^n + b = 57b$, то есть $a \cdot 10^n = 56b$. Левая часть равенства делится на 7, 10^n не делится на 7. Значит, a кратно 7, а, поскольку $a < 10$, то равно 7. После сокращения на 7 получаем $10^n = 8b$. Это равенство возможно при наименьшем $n = 3$ и $b = 125$.

Система оценки

- Полное обоснованное решение — 15 баллов.
- Приведено число, удовлетворяющее условию задачи, но не являющееся наименьшим возможным — 5 баллов.
- Приведен верный ответ, при этом могут быть частичные продвижения, но не доказано, что решение единственное — 5 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: 7125.

Задача III.1.2.2. (15 баллов)

Найдите все положительные решения уравнения

$$x^{2021x} = (2021x)^x$$

Решение

Возведя обе части уравнения в степень $\left(\frac{1}{x}\right)$, получим: $x^{2021} = 2021x$.

Отсюда $x^{2020} = 2021$ и $x = \sqrt[2020]{2021}$.

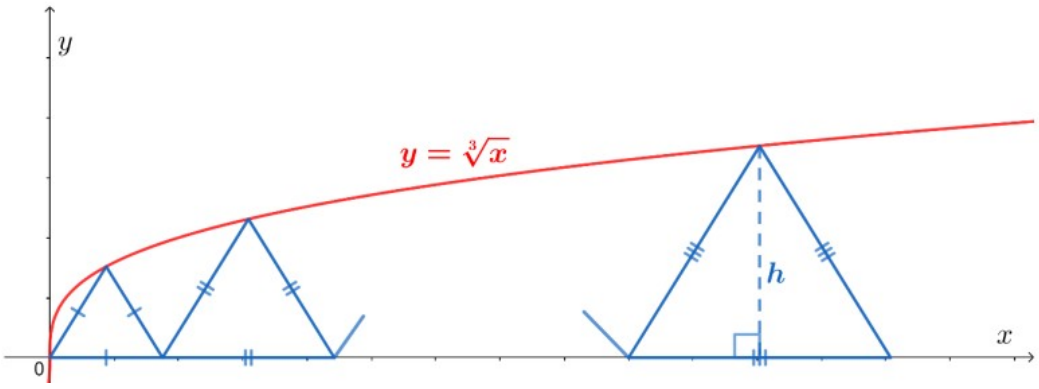
Система оценки

- Полное обоснованное решение — 15 баллов.
- Приведен верный ответ, при этом могут быть частичные продвижения, но не доказано, что решение единственное — 5 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: $\sqrt[2020]{2021}$.

Задача III.1.2.3. (20 баллов)

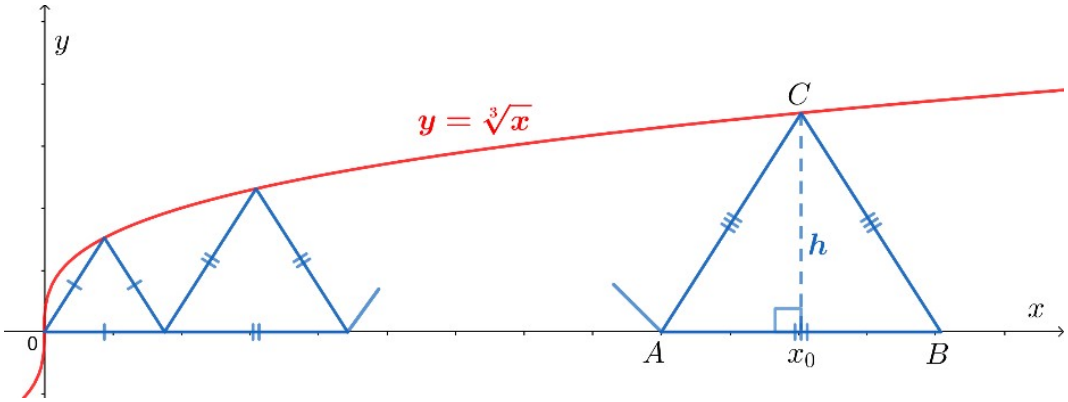
Под графиком функции $y = \sqrt[3]{x}$ при $x \geqslant 0$ построена конечная последовательность равносторонних треугольников (их количество неизвестно), основания которых лежат на оси абсцисс и примыкают друг к другу, начиная от начала координат, а вершины лежат на графике кубического корня. (См. чертеж.) Высота последнего треугольника равна h. Найдите сумму периметров всех треугольников.



Решение

Пусть x_0 — основание высоты последнего треугольника ABC сторона a которого равна $\frac{2h}{\sqrt{3}}$. Тогда $\sqrt[3]{x_0} = h$ то есть $x_0 = h^3$. Значит, точка A имеет координаты $\left(h^3 - \frac{h}{\sqrt{3}}; 0\right)$, то есть сумма всех оснований треугольников кроме последнего равна

$h^3 - \frac{h}{\sqrt{3}}$, а сумма их периметров $3\left(h^3 - \frac{h}{\sqrt{3}}\right)$. Периметр последнего треугольника равен $2\sqrt{3}h$.



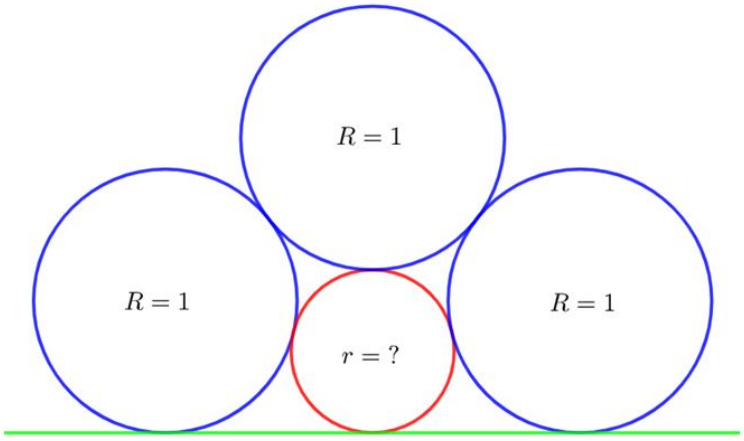
Система оценки

- Полное обоснованное решение — 20 баллов.
- Найдена сторона последнего треугольника — 5 баллов.
- При верном решении получен неверный ответ из-за вычислительной ошибки — 15 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: $3h^3 + h\sqrt{3}$.

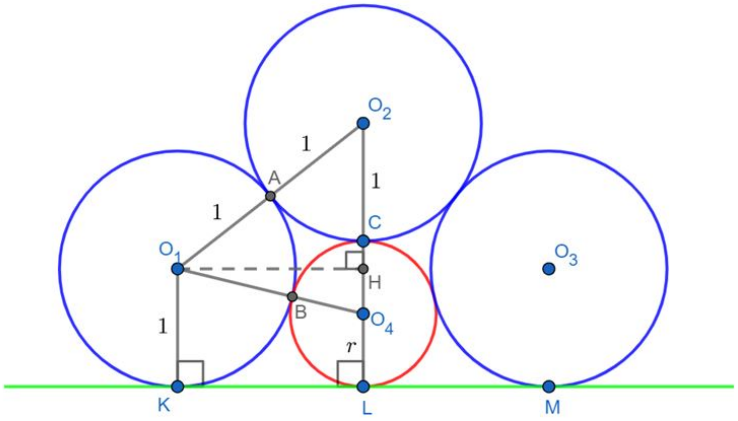
Задача III.1.2.4. (20 баллов)

Радиусы больших окружностей на рисунке равны 1. Две из них касаются прямой и третьей из больших окружностей, а третья касается двух других. Найдите радиус маленькой окружности, которая касается каждой из трех больших окружностей и прямой.



Решение

Обозначим центры больших окружностей O_1, O_2 и O_3, центр маленькой окружности — O_4. A, B, C, K, L и M — точки касания.



1. Из прямоугольного треугольника O_1HO_4 :
   $O_1H^2 = O_1O_4^2 - O_4H^2 = (1+r)^2 - (1-r)^2 = 4r$.
2. $O_2H = O_2C + CH = O_2C + (CL - O_1K) = 1 + 2r - 1 = 2r$.

Запишем теорему Пифагора для ΔO_1HO_2 : $O_1O_2^2 = O_1H^2 + O_2H^2$ или $2^2 = 4r + (2r)^2$.

Единственный положительный корень уравнения $r = \frac{\sqrt{5}-1}{2}$.

Система оценки

- Полное обоснованное решение — 20 баллов.
- Обоснованно получено верное уравнение, при его решении возникли ошибки — 15 баллов.
- Приведен верный ответ, при этом могут быть частичные продвижения — 10 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: $\frac{\sqrt{5}-1}{2}$.

Задача III.1.2.5. (30 баллов)

При каких значениях параметра a уравнение $(a+2)^2x^8 + (a^2+2a)x^5 + 5x - 2 = 0$ имеет единственное решение?

Решение

Решение. При $a = -2$ уравнение имеет единственное решение. При $a \neq -2$ функция:

$$f(x) = (a+2)^2x^8 + (a^2+2a)x^5 + 5x - 2$$

представляет собой многочлен четной степени, значения которого стремятся к $+\infty$ при стремлении x к $+\infty$ и при стремлении x к $-\infty$. В то же время, $f(0) = -2$. Поскольку $f(x)$ — непрерывная функция, то на интервалах $(0; +\infty)$ и $(-\infty; 0)$ функция $f(x)$ принимает нулевые значения, то есть уравнение $f(x) = 0$ имеет, по крайней мере, два различных решения.

Система оценки

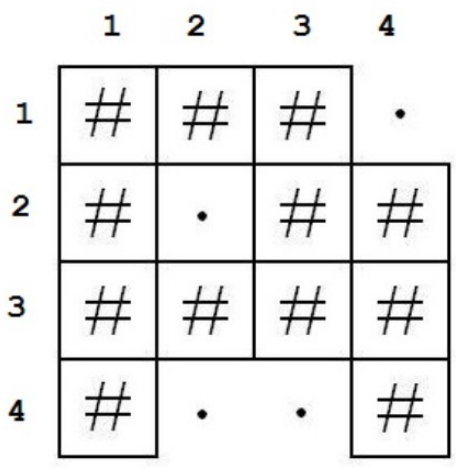
- Полное обоснованное решение — 30 баллов.
- Найден ответ без доказательства, что он единственен — 10 баллов.
- Задача не решена или решена неверно — 0 баллов.

Ответ: $a = -2$.

Информатика. 8–11 класс

Задача III.1.3.1. Шифровальная решетка (15 баллов)

Один из способов шифрования сообщения заключается в применении шифровальной решетки. Из бумаги в клетку вырезается квадрат с четной стороной в $2n$ клеток. При этом в шифруемом сообщении должно быть не более чем $(2n)^2$ символов, включая пробелы. Для построения решетки удалим из нее ровно четверть клеток. При этом должно быть выполнено следующее условие: среди каждых четырех клеток, переходящих друг в друга при повороте квадрата относительно центра, должна быть удалена ровно одна. Например, в качестве экземпляра шифровальной решетки возьмем решетку 4×4, приведенную на следующем рисунке:



Здесь символом «#» обозначены неудаленные клетки, символом «.» обозначены удаленные клетки.

Видно, в частности, что из четырех переходящих друг в друга при повороте клеток (1, 1), (1, 4), (4, 1) и (4, 4) ровно одна на позиции (1,4) отсутствует. Это свойство выполняется и для всех остальных четверок переходящих друг в друга при повороте клеток решетки.

В качестве начального расположения решетки, возьмем изображенное на рисунке, в качестве направления поворота выберем поворот по часовой стрелке, в качестве шифруемого сообщения возьмем сообщение «You are to write». Накладываем решетку на поверхность, желательно в такую же клетку и в пустые клетки вписываем первые четыре символа сообщения, в порядке слева направо сверху вниз (Это часть сообщения «You »). Далее поворачиваем решетку на 90 градусов по часовой стрелке и в новые четыре пустые клетки вписываем следующие четыре символа «are », далее снова поворачиваем и вписываем «to w», и после четвертого поворота последние символы «rite». Если сообщение содержит менее чем $(2n)^2$ символов, то последние символы заменяем на пробелы. После этого убираем решетку и получаем шифровку, приведенную на втором рисунке:

r	t	o	Y
a	o	r	i
e	t		e
w	u		

Дешифровка сообщения происходит так же, как и шифрование, при помощи решетки, которая и является секретным ключом.

Формат входных данных

В первой строке находится четное число K — размер стороны квадрата шифровальной решетки. В следующих K строках содержится описание шифровальной решетки: по K символов в строке, каждый из которых есть либо «#», это означает, что данная клетка не удалена, либо «.», что означает, что данная клетка удалена. Приведенное положение решетки является начальным, поворот при шифровке/дешифровке производится по часовой стрелке. После описания решетки идет зашифрованное сообщение так же из K строк по K символов в каждой, сообщение состоит из больших и малых букв латиницы, цифр, знаков препинания и пробелов. $2 \leqslant K \leqslant 100$.

Формат выходных данных

В единственной строке вывести исходное сообщение длиной K^2 символов, при этом должны быть восстановлены все символы сообщения, включая пробелы, в том числе и в конце сообщения.

Примеры

Пример №1

Стандартный ввод
4 ###. #.## #### #..# rtoY aori et e wu
Стандартный вывод
You are to write

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

int n;
vector<string> R, T;
string ans;

vector<string> rot(){
        vector<string> r = R;

        for(int i = 0; i < n; i++)
            for(int j = 0; j < n; j++)
             r[j][n - 1 - i] = R[i][j];
  return r;
}

void read(){
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
         if(R[i][j] == '.') ans += T[i][j];
         R = rot();
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

        cin >> n;
        R.resize(n);
        T.resize(n);

        for(int i = 0; i < n; i++)
```

```
            cin >> R[i];
            getline(cin, T[0]);
        for(int i = 0; i < n; i++)
           getline(cin, T[i]);

        for(int i = 0; i < 4; i++)
            read();

        cout << ans<<endl;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
n = int(input())
res = []
for i in range(n):
    ff = input()
    sd = []
    for el in ff:
        sd.append(el)
    res.append(sd)
dish = []
for i in range(n):
    asd = input()
    aa = []
    for el in asd:
        aa.append(el)
    dish.append(aa)
ans = ''
for step in range(4):
    for i in range(n):
        for j in range(n):
            if res[i][j] == '.':
                ans += dish[i][j]
    new = [[0] * n for i in range(n)]
    for i in range(n):
        for j in range(n):
            new[j][n - 1 - i] = res[i][j]
    res = new
print(ans)
```

Задача III.1.3.2. Разработка и тестирование системы Диффи – Хеллмана (15 баллов)

Первой разработкой в области криптосистем с открытым ключом считается система Диффи – Хеллмана. В этой задаче проведем ее разработку и тестирование для заданных параметров.

Система основана на результатах теории чисел. Для ее использования требуется выбрать большое простое число p и некоторое число d, такое, что $1 < d < p - 1$.

Числа p и d публикуются на странице пользователей криптосистемы. Далее каждый желающий присоединиться к этой системе пользователь A выбирает случайное

большое число T_A и вычисляет величину $Z_A = d^{T_A} \ mod \ p$. Далее напротив своего имени в таблице пользователей он указывает это число Z_A, которое становится известно всем желающим и называется открытым ключом пользователя A. При правильном подборе числа T_A, по числам Z_A, p и d восстановить T_A за разумное время не получится. Таким образом, числом T_A обладает только пользователь A. Аналогично поступает и второй пользователь B.

Допустим, теперь пользователь A хочет отправить секретное сообщение пользователю B. Для их приватного общения им нужно договориться о секретном ключе. Это одно большое число K_{AB}, которым должны обладать только A и B и никто другой. До появления криптосистем с открытым ключом этот ключ нужно было как-то передавать от одного к другому. Теперь же для вычисления K_{AB} нужно сделать следующее: пользователь A берет число Z_B и вычисляет $(Z_B)^{T_A}$. А пользователь B вычисляет $(Z_A)^{T_B}$. Согласно теории чисел, при правильном подборе p и d, эти две величины должны совпасть, а значит, пользователи A и B получили ключ K_{AB}, который не пересылался по каналам связи и может быть получен только этими пользователями. Кроме того, ключ K_{AB} нигде не хранится, что так же повышает надежность системы.

Рассмотрим следующий простой пример: $p = 47$, $d = 19$.

Теперь пусть $T_A = 40$, тогда $Z_A = 21$. А T_B пусть равно 28, тогда $Z_B = 2$.

Пользователь A вычисляет $(Z_B)^{T_A} mod \ p = 2^{40} mod \ 47 = 36$.

А пользователь B вычисляет $(Z_A)^{T_B} mod \ p = 21^{28} mod \ 47 = 36$.

Таким образом $K = 36$.

По заданным параметрам p и d найти для n пользователей по их закрытым ключам T_i их открытые ключи Z_i и далее для каждой пары i, j найти их общий ключ K_{ij} и проверить совпадение.

Формат входных данных

В первой строке содержатся два числа p и d ($2 \leqslant d < p - 1 < 10^{18}$), p — простое.

Во второй строке находится число n — количество пользователей криптосистемы. $2 \leqslant n \leqslant 100$.

В следующих n строках находятся числа T_i — секретные ключи пользователей. Ключ пользователя номер i содержится в строке номер $i + 2$ ($2 \leqslant T_i \leqslant p - 1$).

Формат выходных данных

В первых n строках вывода указать числа Z_i в порядке, соответствующем входу. Далее в $n \cdot (n - 1)/2$ строках вывести пары чисел $(Z_i)^{T_j}$ и $(Z_j)^{T_i}$ через пробел для всех пар различных пользователей i, j. Пары выводить в порядке возрастания первой компоненты i, а при равенстве первой — в порядке возрастания второй компоненты j.

В последней строке вывести «ok», если в каждой паре происходит совпадение ключей или «bad», если есть хотя бы одна пара, где совпадения нет.

Примеры

Пример №1

Стандартный ввод
47 19 3 40 28 33

Стандартный вывод
21 2 43 36 36 27 27 37 37 ok

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

ll binprod (ll a, ll b, ll p) {
    ll res = 0;
    while (b) {
        if (b & 1)
            res = (res + a) % p;
        a = (a + a) % p;
        b /= 2;
    }
    return res;
}

ll binpow (ll a, ll n, ll p) {
    ll res = 1;
    while (n) {
        if (n & 1)
            res = binprod(res, a, p);
        a = binprod(a, a, p);
        n >>= 1;
    }
    return res;
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    bool ok = 1;
```

```
    ll p, d, n;
    cin >> p >> d >> n;
    vector<ll> T(n);
    for(int i = 0; i < n; i++)
        cin >> T[i];
    vector<ll> Z(n);
    for(int i = 0; i < n; i++)
        Z[i] = binpow(d, T[i], p);
    for(int i = 0; i < n; i++)
        cout<<Z[i]<<endl;

    for(int i = 0; i < n-1; i++)
        for(int j = i + 1; j < n; j++){
            ll ij = binpow(Z[i], T[j], p);
            ll ji = binpow(Z[j], T[i], p);
            cout << ij <<' '<< ji<<"\n";
            if(ij != ji) ok = 0;
        }
       cout<< (ok? "ok" : "bad")<<endl;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
def bin_pow(a, n, p):
    if (n == 0):
        return 1;
    if (n % 2 == 0):
        b = bin_pow(a, n // 2, p)
        return (b * b) % p
    return (a * bin_pow(a, n - 1, p) ) % p
ff = input().split()
p = int(ff[0])
d = int(ff[1])
n = int(input())
mas = []
for i in range(n):
    mas.append(int(input()))

for i in range(n):
    print(bin_pow(d, mas[i], p))
for i in range(n):
    for j in range(i + 1, n):
        pr = bin_pow(d, mas[i] * mas[j], p)
        print(pr, pr)
print('ok')
```

Задача III.1.3.3. Ёмкость кода (15 баллов)

Начинающий криптоаналитик Вениамин разработал новый способ кодирования информации, название которому он пока не придумал. Принцип кодирования секретен, но известно, что он основан на числах, десятичная запись которых не содержит двух рядом стоящих цифр таких, что одна из них четная, а вторая кратна трем. Например, число 12345667890 использовать нельзя, так как в нем соседствуют цифры

23, 34, 66, 89 и 90 – каждая такая пара запрещена. А, например, число 10524847339707 вполне допустимо.

Очевидно, что количество шифруемых этим кодом состояний (его емкость) зависит от количества разрешенных чисел. И теперь Константина Поликарповича – научного руководителя Вениамина — заинтересовал вопрос: а сколько есть разрешенных для кодирования чисел, запись которых состоит из n цифр и не содержит ведущих нулей.

Формат входных данных

На вход подается единственное число n. $1 \leqslant n \leqslant 22$.

Формат выходных данных

В ответ вывести одно число – количество разрешенных чисел длины n без ведущих нулей, больших либо равных 1.

Примеры

Пример №1

Стандартный ввод
3
Стандартный вывод
446

Пример программы-решения

Ниже представлено решение на языке C++.

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


using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    ll n, m = 1, rA = 5, rB = 4, a0 = 1, b0 = 1, r0 = 0, p0 = 1;
    cin >> n;

    ll st = 1;
    for(int i = 0; i < m; i++)
        st *= 10LL;

    ll p = 2;
    a0 -= p0;
    b0 -= p0;
```

```
22
23      ll r = st - rA - rB + p;
24      r0 = st/10 - a0 - b0 - p0;
25
26      rA -= p;
27      rB -= p;
28
29      vector<vector<ll> > dp(4, vector<ll>(n+1, 0));
30
31      dp[0][0] = 1;
32      if(m > 1){
33          dp[0][1] = 9;
34          for(int i = 2; i < m; i++) dp[0][i] = dp[0][i-1] * 10;
35      }
36
37      dp[0][m] = r - r0;
38      dp[1][m] = rB - b0;
39      dp[2][m] = rA - a0;
40      dp[3][m] = p - p0;
41
42      for(int i = m+1; i <= n; i++){
43          dp[0][i] = (dp[0][i-m] + dp[1][i-m] + dp[2][i-m] + dp[3][i-m]) * r;
44          dp[1][i] = (dp[0][i-m] + dp[1][i-m]) * rB;
45          dp[2][i] = (dp[0][i-m] + dp[2][i-m]) * rA;
46          dp[3][i] = dp[0][i-m] * p;
47      }
48
49      ll ans = 0;
50      for(int i = 0; i < 4; i++)
51          ans += dp[i][n];
52
53      cout << ans;
54  }
```

Задача III.1.3.4. Экономный криптоаналитик (25 баллов)

В теории защиты информации огромную роль играют так называемые полупростые бесквадратные числа. Это числа, которые являются произведением двух различных простых чисел. Например, число 2021 является произведением двух различных простых чисел 43 и 47. Поиск больших полупростых чисел является достаточно трудоемким занятием, поэтому некоторые фирмы специализируются на разработке и продаже наборов полупростых чисел для нужд криптоаналитиков.

В нашей задаче начинающий криптоаналитик Вениамин задумался о покупке большого набора полупростых чисел. Для пробы он хочет купить набор маленьких полупростых чисел, чтобы самому потренироваться в их разложении на простые делители. В его понимании «маленькими» являются числа от 2 до 10^7. В данный период действует акция: каждое число в наборе можно купить независимо от других, при этом все числа в наборе стоят одинаково. Несмотря на то, что Вениамин является начинающим криптоаналитиком, он уже знает, что из одних полупростых чисел можно бесплатно получать другие по следующему правилу: пусть у нас в собственности есть два полупростых числа $A = p_1 \cdot p_2$ и $B = p_2 \cdot p_3$, тогда можно бесплатно получить третье полупростое число $C = p_1 \cdot p_3$. Это новое число C можно далее использовать в аналогичной конструкции для получения других полупростых чисел.

Вениамин хочет стать владельцем всех чисел из набора и при этом минимизиро-

вать свои затраты. По этой причине он хочет некоторые числа из набора не покупать, получив их из других купленных, по указанному выше правилу. Теперь нужно выяснить, сколько чисел в предложенном Вениамину наборе можно не покупать, а получить бесплатно.

Формат входных данных

В первой строке содержится количество чисел n в предлагаемом наборе. $1 \leqslant n \leqslant 10^5$.

Во второй строке содержатся n попарно различных чисел, которые предложены Вениамину. Любые два из них разделены ровно одним пробелом. Вениамин может выбирать покупать ему или нет очередное число. Фирма гарантирует, что все предложенные числа полупростые бесквадратные. Каждое из чисел находится в пределах от 2 до 10^7.

Формат выходных данных

В ответ вывести одно число — количество таких чисел среди исходных, которые можно получить из других по указанному выше правилу, и, соответственно, не покупать.

Пояснение к примеру 1.

Нам даны 15 чисел. Покажем, что 4 из них можно бесплатно получить из остальных 11.

Купим числа $6493 = 151 \cdot 43$ и $5977 = 43 \cdot 139$, тогда число $20989 = 151 \cdot 139$ можно получить из двух уже купленных.

Далее купим число $4309 = 31 \cdot 139$ и добавив к нему $20989 = 139 \cdot 151$ получим $4681 = 31 \cdot 151$. Этого числа в исходном наборе нет, но мы его и не покупаем, а получаем бесплатно.

Теперь купим число $6493 = 43 \cdot 151$ и добавим к нему $4681 = 151 \cdot 31$, тогда бесплатно получим $1333 = 43 \cdot 31$, его тоже нет в исходном наборе.

Теперь осталось купить число $1457 = 47 \cdot 31$ и добавить к нему $1333 = 31 \cdot 43$ и мы получим второе число из набора $2021 = 47 \cdot 43$.

Теперь купим числа $3151 = 137 \cdot 23$ и $253 = 23 \cdot 11$ и мы получим еще одно число из набора $1507 = 137 \cdot 11$.

Осталось купить число $5497 = 239 \cdot 23$ и добавить к нему $253 = 23 \cdot 11$, и мы получим еще одно число из набора $2629 = 239 \cdot 11$.

Таким образом, мы получили четыре числа из набора бесплатно (20989, 2021, 1507 и 2629).

Можно показать, что остальные числа из набора придется покупать непосредственно. Стоит сказать, что приведенный вариант покупки не единственный, но в любом случае 4 из предложенных 15 чисел можно получить бесплатно.

Примеры

Пример №1

Стандартный ввод
15 2021 20989 253 4309 1891 1457 1507 5977 6493 3763 2491 1385 3151 2629 5497
Стандартный вывод
4

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;

const int MAX = 1e7;

int get_par(int u, vector<int> &p){
    if (p[u] == u)
        return u;
    return u = get_par(p[u], p);
}

void uni_on(int u, int v, vector<int> &p, vector<int> &d){
    int p1 = get_par(u, p);
    int p2 = get_par(v, p);
    if (p1 == p2)
        return;
    if (d[p1] < d[p2])
        swap(p1, p2);
    p[p2] = p1;
    if (d[p1] == d[p2])
        d[p2]++;
}

int main(){
    ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
    int n;
    cin >> n;
    vector<vector<int>> check(5000001, vector<int>());
    vector<int> divers(1e7 + 1, -1);
    for (int i = 2; i <= MAX; ++i){
        if (divers[i] == -1){
            int k = i;
            divers[i] = k;
            for (k; k <= MAX; k += i){
                if (divers[k] == -1)
                    divers[k] = i;
            }
        }
    }
    vector<int> p(1e7 + 1);
    vector<int> d(1e7 + 1, 1);
```

```
    for (int i = 1; i <= 1e7; ++i)
        p[i] = i;
    int ans = 0;
    for (int i = 0; i < n; ++i){
        int a;
        cin >> a;

        vector<int> div;
        div.push_back(divers[a]);
        a /= divers[a];
        div.push_back(a);
        if (get_par(div[0], p) == get_par(div[1], p)){
            ans++;
            continue;
        }
        uni_on(div[0], div[1], p, d);
    }
    cout << ans;
}
```

Пример программы-решения

Ниже представлено решение на языке Python.

```
def get_par(u):
    global p
    if p[u] == u:
        return u
    u = get_par(p[u])
    return u
def uni_on(u, v):
    global p
    global d
    p1 = get_par(u)
    p2 = get_par(v)
    if p1 == p2:
        return
    if d[p1] < d[p2]:
        AS = p1
        p1 = p2
        p2 = AS
    p[p2] = p1
    if d[p1] == d[p2]:
        d[p2] += 1
MAX = 10000000
n = int(input())
a = list(map(int, input().split()))
p = [0] * 10000005
d = [0] * 10000005
for i in range(MAX + 1):
    p[i] = i
ans = 0
for i in range(n):
    num = a[i]
    k = 2
    div = []
    while k * k <= num:
        if num % k == 0:
            div.append(k)
```

```
            num = num // k
            break
        k+= 1
    div.append(num)
    if get_par(div[0]) == get_par(div[1]):
        ans += 1
        continue
    uni_on(div[0], div[1])
print(ans)
```

Задача III.1.3.5. Докторская Константина Поликарповича (30 баллов)

Константин Поликарпович – научный руководитель начинающего — криптоаналитика Вениамина очень заинтересовался открытием своего ученика. Само собой, он сразу же понял, что код Вениамина из задачи 3 – это только частный случай очень обширного семейства защитных кодов. Константин Поликарпович проделал очень серьезное исследование для обобщения этих кодов и его результат представил в своей закрытой докторской диссертации. По известным причинам, большинство ее пунктов нам не доступны, однако некоторые детали носят открытый характер. В частности известно, что при обобщении кодов Вениамина выделен следующий способ построения разрешенных для кодирования чисел длины n без ведущих нулей:

выбирается число m – длина опорного блока, $1 \leqslant m \leqslant 3$. Опорным блоком называется набор цифр длины m (он может начинаться даже с нуля). Зафиксируем два набора таких блоков A и B (наборы могут пересекаться). Число длины n считается разрешенным для кодирования, если в его записи нигде нет двух рядом стоящих блоков, один из которых из множества A, а другой из множества B. Вопрос остается тем же, что и в предыдущей задаче – по заданным n, A, B подсчитать количество разрешенных чисел (емкость кода с указанными параметрами).

Формат входных данных

В первой строке находятся четыре числа n, m, rA, rB через пробел, $1 \leqslant n \leqslant 19$, $1 \leqslant m \leqslant 3$, $1 \leqslant rA, rB \leqslant 10^m$. Числа rA и rB — длины наборов A и B соответственно. Во второй строке содержится набор блоков A через пробел, в третьей строке содержится набор блоков B. Длина каждого блока ровно m цифр. Внутри одного набора блоки попарно различны.

Формат выходных данных

В ответ вывести одно число — количество разрешенных чисел длины n без ведущих нулей, больших либо равных 1.

Пояснение к примеру 1.

В первом примере представлен случай «кода Вениамина» из задачи C. Действительно, в качестве множества A выступает множество четных цифр, в качестве множества B выступает множество цифр, кратных трем. Ответ, очевидно, должен совпадать с ответом на аналогичный тест из той задачи.

Примеры

Пример №1

Стандартный ввод
3 1 5 4 0 2 4 6 8 0 3 6 9
Стандартный вывод
446

Пример №2

Стандартный ввод
5 2 4 3 01 00 13 43 01 10 04
Стандартный вывод
89696

Пример программы-решения

Ниже представлено решение на языке C++.

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
typedef long double ld;

int len(int a){
    int r = 0;
    while(a){
        r++;
        a /= 10;
    }
    return r;
}

int main(){
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

    ll n, m, rA, rB, a0 = 0, b0 = 0, r0 = 0, p0 = 0;
    cin >> n >> m >> rA >> rB;
    vector<int> A(rA), B(rB);

    for(int i = 0; i < rA; i++) {
            cin >> A[i];
    }

    for(int i = 0; i < rB; i++) {
            cin >> B[i];
    }
```

```
31
32      ll st = 1;
33      for(int i = 0; i < 2*m-1; i++)
34          st *= 10LL;
35
36      ll mst = 1;
37      for(int i = 0; i < m; i++)
38          mst *= 10LL;
39
40      vector<vector<bool> > msk(mst, vector<bool>(mst, 1));
41
42      for(int i = 0; i < A.size(); i++)
43          for(int j = 0; j < B.size(); j++){
44              msk[A[i]][B[j]] = 0;
45              msk[B[j]][A[i]] = 0;
46          }
47
48      vector<vector<ll> > dp(st, vector<ll>(20, 0));
49
50      for(int i = 0; i < st; i++){
51          dp[i][len(i)] = 1;
52      }
53
54      for(int u = 2*m-1; u < n; u++){
55          for(int i = 0; i < st; i++) if(dp[i][u] > 0){
56              for(int k = 0; k < 10; k++) {
57                  int a = i * 10 + k;
58                  int b = a % mst;
59                  int c = a / mst;
60                  if(msk[c][b]){
61                          int d = a % st;
62                      dp[d][u+1] += dp[i][u];
63                  }
64              }
65          }
66      }
67
68      ll ans = 0;
69      for(int i = 0; i < st; i++) ans += dp[i][n];
70      cout<<ans;
71  }
```