Работы победителей и призеров
Олимпиады школьников "Надежда энергетики" по предмету "информатика"
в 2016/2017 учебном году

Олимпиада школьников «Надежда энергетики»

МЭИ

Место проведения

ЙR86-61

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ БЕЛАВЕНЦЕВ

ИМЯ ВАЛЕРИЙ

ОТЧЕСТВО ЕВГЕНЬЕВИЧ

Дата рождения 20.05.1999

Класс: 11

Предмет ИНФОРМАТИКА

Этап: ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 8 **листах**

Дата выполнения работы: 18.02.2017
(число, месяц, год)

Подпись участника олимпиады:

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ UR86-61

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№1.

Считаем числа P, Q, B.

Чтобы достаточно эффективно найти все простые числа ~~от~~ до B (включительно), воспользуемся решетом Эратосфена.

Строится оно следующим образом:

Создаем массив на B+1 элемент (с индексами от 0 до B включит).

Для удобства положим, что a[i] отвечает за число i (a - назв. массива).

Тогда изначально положим a[0]=0, a[1]=0, ~~a[2]~~, ~~a для i = 2..~~ а все ост. элементы положим a[i] = 1.

* Если a[i]==0 будем называть число зачеркнутым, если a[i]==1 будем наз-ть незачеркнутым.

Тогда алгоритм следующий: циклом от 2 до B (включит) делаем:

находим ~~[illegible]~~ наименьшее незачеркнутое число j такое, что j≥i (i - текущая итерация цикла). Все элементы массива вида a[j·n], где n ∈ N и n>1, зачеркиваем (приравниваем к 0).

Тогда после выпол. цикла от 2 до B незачеркнутые числа и будут простыми. (см. след. стр)

№ 1 (прод)

Для удобства перенесем ~~эти засорунты~~ эти прост. числа в другой массив p. (1)

P.S. Да, построение решета можно несколько ускорить, рассматривая, например, только нечет. числа (ведь все чет. числа кроме 2 составные), но тогда будет несколько менее удобно работать с индексами массива.

Вернемся к основной задаче. Далее все нетрудно: перебираем каждое число от P до Q. Для каждого из чисел переберем все простые числа из массива p. (который мы получили после решета) и проверим,

(*) если число делится на прост. число и если это прост. число > B, то это число не является B-гладким. ~~Иначе~~

если условие (*) ни разу не выполнилось для числа x => число x B-гладкое.

P.S. Алгоритм можем еще улучшить.

Понятно, что интересно рассматривать только простые числа > B. Тогда в массив p (1) можно заносить из решета только простые числа > B.

0

(+)

см. след лист.

№2.

P.S. Я искал именно y, а не x, т.к. $R_{257}(a^x) < R_{257}(a^y) \Rightarrow$ хочется искать ост. от дел. более маленького числа в степени

Девочки знают число a.
Т.к. $R_{257}(a^i)$ различны $\Rightarrow$ зная $R_{257}(a^y) = 256$ можно однозначно определить y.

Определять мы будем перебором всех i от 1 до 256 (включительно), пока не встретим такое i, что $R_{257}(a^i) == 256$.

Чтобы не хранить большие числа в памяти компьютера можно воспользоваться следующим свойством:
Если $R_{257}(x) == k$, то $R_{257}(x^y) == R_{257}(k^y)$
(св-во перемножения остатков (*)
Это позволит нам при переборе не возводить число a в большие степени.

Зная y, также воспользуемся свойством:
$R_{257}(a^x) == 9 \Rightarrow R_{257}(a^{xy}) == R_{257}(9^y)$.
Таким образом, зная y, мы без труда найдем $R_{257}(9^y)$. Опять же, пользуясь свойством перемножения остатков, мы сможем не хранить число в большой степени в памяти компьютера.

(*) Св-во перемножения остатков выглядит следующим образом:

$$\begin{cases} a \equiv c \pmod{k} \\ b \equiv d \pmod{k} \end{cases} \Rightarrow ab \equiv cd \pmod{k}$$

№3.

Считаем матрицу в виде двумерного массива размером n×n.

Создадим двумерный массив B размера n×n. (это будет нашей новой искомой матрицей)

Тогда алгоритм понятен: двумя вложенными циклами от 1 до n (включительно, будем считать, что нумерация индексов в массиве начинается с 1.) будем перебирать каждую клетку массива B и в нее записывать нужное нам значение (тот самый минимум) из массива A.

Покажем теперь, как найти этот самый минимум. Пусть у нас на какой-то итерации ~~Создади~~ вложенные циклы дают числа i и j - координаты очередной клетки в таблице B.

~~Создад~~ Создадим переменную, в которой и будет наш минимум (переменная m);
переменная → level=1
изначально m = A[i][j] (* = - присвоение, == - сравнение - обозначение псевдокода).

Пройдемся циклом k от i+1 до n - включительно.
На каждой итерации цикла обновляем переменные left и right, level.

~~left = max(~~ ~~left = max(1, j-level~~

~~left = max(1, j-level)~~ см. след. стр.
~~right = min(n, j+level)~~
~~level = level+1~~

~~(см. след. стр)~~

(*) ~~max (a, b) – максимум из чисел a, b~~
~~min (a, b) – минимум из чисел a, b~~
~~обозначения псевдокода.~~

~~Это необходимо, чтобы обработать~~

left = j - level
right = j + level
level = level + 1

~~Затем обновляем переменную m:~~
~~m = max (m, a[k][~~

Затем, пока left <= right делаем следующее: проверяем, входит ли клетка A[k][left] в границы массива A.

Если входит, то обновляем максимум:
* m = max(m, a[k][left])
left = left + 1 – увелич. left в любом случае

После выполнения цикла k присвоим ~~независимо от того~~
B[i][j] = m.

а это все внутри условия „пока"

это все внутри цикла k

Таким образом мы сможем построить нужную нам матрицу B.

* max (a, b) – максимум из чисел a и b – обозначение псевдокода

(см. след. стр.)

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№4.

Нетрудно понять, что для успешного выполнения условия нам достаточно научиться менять местами два элемента в таблице. Покажем, как это делать. Пусть есть двумерный массив a. Пусть мы хотим в нем поменять местами элементы a[m][n] и a[i][j].

Для этого сделаем следующее:

```
k = a[m][n]
a[m][n] = a[i][j]
a[i][j] = k
```

(„=" - присвоение
„==" - сравнение)

Хорошо, менять местами элементы мы научились. Покажем же теперь ~~как именно~~ в каком именно порядке ~~[illegible]~~ надо менять элементы. i↓ →j

1	2
4	3

Сначала поменяем местами первые и третьи куски.

```
* for i in range(n):  - цикл i от 0 до n-1 включ.
      for j in range(n):  - цикл j от 0 до n-1 включ
          swap(a[i][j], a[i+n][j+n])
```

меняем местами эл-ты a[i][j] и a[i+n][j+n].
(как менять описано в начале)

(*) пусть ~~эл-ты~~ индексы в таблице начинаются с 0 - для удобства.

Теперь поменяем местами второй и четвертый куски.

(см. след. стр)

~~for i in range(n, 2n):~~
~~for i in range(~~

for i in range(n): – цикл i от 0 до n-1 включ.
for j in range(n, 2n): – цикл j от n включит до 2n не включительно
swap(a[i][j], a[i+n][j-n])

меняем местами эл-ты
a[i][j] и a[i+n][j-n].
(как менять описано в начале).

После этого задача будет решена.

№5.

Т. к. про нижнюю границу ничего не сказано, а давать ограничение только по верхней границе - бессмысленно, то примем нижнюю границу как $n_1 = -10^{15}$.

Тогда будем для чисел от -10^{15} до 10^{15} перебирать все возможные последовательности длины $2 \cdot 10^{15}+1$, состоящие из 0 и 1, где 0 значит, что мы не "берем" число i (a[i]=0), а 1 означает, что мы "берем" число i в наше рассматриваемое множество.

(*) Для удобства будем хранить каждую такую последовательность в неком массиве a, причем индексация у этого массива начинается с -10^{15}.

(см. след. стр.)

№5 (прод.)

(**)

Если такой возможности нет (напр. язык программирования не поддерживает отрицательную индексацию) => просто будем считать что элемент массива a[i] отвечает за $i - 10^{15}$ число

Для каждой полученной последовательности будем проверять условия:

1. кол-во единиц в ней лежит в промежутке от P до Q
2. выполняется условие про кажд. число в множестве... (см. условие)

Как проверить первое усл. - очевидно - линейным проходом берем и считаем кол-во единиц

Покажем как наиболее эффективно проверить второе условие.

Для этого в некоторую переменную „m" запишем произвед. всех чисел, входящих в последовательность, n_i последовательности

Затем для каждого числа будем делать следующее: проверим, делится ли число $(m / n_i + 1)$ на n_i. Если не делится, то такое множество нам точно не подходит. Если мы перебрали все числа и все время у нас делилось, то такое множество нам подходит => увеличиваем переменную, отвечающую за число решений на 1.

P.S. число „m" может получиться достаточно большим, поэтому его надо хранить как массив цифр, а операции умножения и деления производить „в столбик"

Олимпиада школьников «Надежда энергетики»

МЭИ

Место проведения

ЙR86-51

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ Бетродова

ИМЯ Берта

ОТЧЕСТВО Ермаковна

Дата рождения 18.01.2000

Класс: 11

Предмет информатика

Этап: заключительный

Работа выполнена на 6 **листах**

Дата выполнения работы: 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Бер

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

1.

Пройдём циклом по всем числам от p до q.
Будем разбивать каждое число на простые множители, до тех пор, пока:
а) не разбили число полностью
б) нашелся делитель, больший B.
Пусть на вход подаются числа p, q и B. Программа выведет все B-гладкие числа в диапазоне от p до q. (код на C++ с коммент.)

```
int main()
{
  int p, q, B;
  cin >> p >> q >> B;   // ввод чисел.
  for (int i = p; i ≤ q; ++i)   // проходимся циклом по всем числам диапазона
  {
    int t = i;      // сохраним i для дальнейшего возможного уменьшения
    int div = 2;    // делитель числа на данной итерации
    bool f = 1;     // флаг, определяющий, что нет делителя, большего B.
    while (t > 1 && f)
    {
      if (t % div == 0)   // проверка на делимость
      {
        if div > B
          f = 0;
        else
          t = t / div;   // сокращаем обрабатываемое t на делитель
      }
      else
        ++div;
    }
    if (f)
      cout << i << endl;   // f = 1 только если в разбиении нет делителей, больших B.
  }
  return 0;
}
```

(+)

2.

Пусть есть число a.

Пусть $a \% 257 = r_0$ (где $a \% x$ - остаток от деления a на x)

Пусть $a^2 \% 257 = r_1$.

С другой стороны, $a \% 257 = r_0 \Leftrightarrow a = 257t_0 + r_0$ ($t_0 \in \mathbb{Z}$)

$a^2 \% 257 = r_1 \Leftrightarrow a^2 = 257t_1 + r_1$ ($t_1 \in \mathbb{Z}$)

$a^2 = a \cdot a = (257t_0 + r_0) \cdot a = 257t_0 + \underline{r_0 \cdot a}$, т.е. $\underline{r_1 = (r_0 \cdot a) \% 257}$.

Соответственно, свойство 1

$a^3 \% 257 = (r_0 \cdot a^2) \% 257 = (r_1 \cdot a) \% 257$ ~~и т.д.~~, и т.д.

Нам известно, что $a^{256} \% 257 = 1$, $a \leq 256$, т.е. перебор не большой. Также надо проверять, чтобы не попадались одинаковые остатки ~~деления~~. Остатки не превосходят 256, поэтому заведём массив на 257 элементов, в котором для каждого a будем подсчитывать кол-во остатков, ~~для~~ равных i-ому эл-ту массива. (код на С++ с комментариями)

```
int main()
{ int A; // переменная, куда сохранится найденное a
  for (int a=2; a <= 256; ++a)
  {
    int mas[257]={0}; // массив для подсчёта остатков.
    bool f=1; // флаг, определяющий, что пока не нашлось одинаковых остатков
    int r_pr=a; // переменная, хранящая остаток от деления на предыдущей итерации;
    mas[r_pr]=1
    for (int st = 2; st <= 256; ++st)
    {
      r_pr=(r_pr*a)%257; (по св-ву 1)
      ++mas[r_pr];
      if mas[r_pr]>1
        f=0; // если ~~так~~ уже был остаток ~~деления~~, равный r_pr.
    }
    if (f && r_pr == 1)
      ~~cout << a~~
      A=a;
  }
```

АЛГОР-М ≠ ПРОГР-МА

~~Таким образом, программа выведет число a.~~

Таким образом, в A сохранится значение загаданного Мариной числа

(2. продолжение)

Найдём теперь число y. Т.к. a^i даёт различные остатки при делении на 257, число y единственно для данного a

~~int main()~~

~~{~~ ... (продолжение программы)

```
int y = 1; int A1 = A; //сохраним A в A1 для изменения;
while (A1 != 256)
{ ++y;
  A=(A1*A)%257;
}
```

Теперь имеем числа a и y. Нужно найти $(a^{xy})\%257 \Leftrightarrow$

~~$a^{xy} = (a^x)^y$~~ $\Leftrightarrow a^{xy} = 257t + r$ $(t \in \mathbb{Z}, r \in \mathbb{N})$, нужно найти r.

Мы знаем, что $a^x \% 257 = 9$, т.е. $9^y = 257t + r$.

~~З~~ Мы знаем y, таким образом, $r = 9^y \% 257$

```
... (продолжение программы)
int ch = 1; //переменная, в которой хранится остаток от деления 9^i на 257
for (int i = 1; i <= y; ++i)
{
   ch *= 9;
   ch %= 257; // по св-ву 1 можем так сделать
}
cout << ch;
return 0;
}
```

Программа выведет секретное искомое число

№ 3.

Будем идти циклами по строкам и столбцам исходной матрицы. Для каждой клетки будем проходиться циклом по соответствующей закрашенной области и находить максимум. Код на C++ с комментар. (пусть ~~а и~~ исходная матрица имеет размер n×n)

```
int main()
{ int A[n][n]; // предполагается, что A уже задана;
  int B[n][n];
  for (int I=0; I<n; ++I) //основной цикл по строкам
    for (int J=0; J<n; ++J) //основной цикл по столбцам
    { B[I][J] = -1e9; // предположим, что меньших чисел в матрице A быть не может
      for (int i=I; i<n; ++i) //цикл по строкам закрашенной области (если граничные диагональные клетки не входят в закраш. область, то с i = I+1)
        for (int j = max(0, J-(i-I)); j<=J+(i-I); ++j) // цикл по столбцам закр. области; если граничные клетки не входят в обл-ть, то с j = max(0; J-(i-I)+1) по j < min(n; J+(i-I))
          if (A[i][j] > B[i][j])
            B[i][j] = A[i][j];
    }
  return 0;
}
```

на и начало

```
for (int j = max(0, J-(i-I)); j <= min(n-1; J+(i-I)) ++j)
```

(+)

№4.

1	2
3	4

Поменяем сначала блоки 1,4,
потом 2,3.

(код на C++ с комментариями)

Алгм ≠ прогр

+

```
int main()
{
  for (int i=0; i<n; ++i)      // идём циклом по строкам и столбцам
    for (int j=0; j<n; ++j)    // 1 блока.
                               // соответствующая клетка 4 блока:
                               // {i+n, j+n}
      swap(a[i][j], a[i+n][j+n]); // меняем клетки местами
  for (int i=0; i<n; ++i)
    for (int j=n; j<2n; ++j)   // идём циклом по строкам и столбцам
                               // 2 блока.
                               // соответствующая клетка 3 блока:
                               // {i+n, j-n}.
      swap(a[i][j], a[i+n][j-n]);
}
```

Сложность (временная) ~ $O(n^2)$

№5.

Имеем k позиций. Будем рекурсивно перебирать числа из интервала от p до q на каждую из позиций. Заведём массив "set", ~~rez~~ в котором на i-ой позиции будет стоять выбранное на данной итерации число. Если числа в выборке не могут повторяться, заведём массив used [q-p], в i-ой ячейке которого хранится инф-ия, использовано ли число [q-p+i]. В коде эти строки будут закомментированы, т.к. возможно, что числа в выборке повторяться могут. (код на C++ с комментариями)

5 (продолжение)

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

using namespace std;
long long p,q,k;
vector<long long> sett;
bool used[q-p];

void rek (long long pr) //pr насчитанное произведение
{
  if sett.size()==k
  {
    bool f=1;  //флаг, показывающий, нашлось ли число в выборке,
               //которое не делится на пр-ие остальных чисел
    for (int i=0; i<sett.size(); ++i)
      if ((pr+1)%sett[i]^2)!=0 || pr+1 sett[i] == sett[i])
        f=0;   //в pr хранится произведение всех чисел из выборки.
               //Таким образом, нужно, чтобы pr/sett[i] % sett[i] = 0,
               //равносильно p/sett[i]^2 = 0)
    if (f)
    {for (int i=0; i<sett.size(); ++i)
       cout<<sett[i]<<' ';
     cout<<endl;   // если выборка подходит, выводим ее.
    }
  }
  else
  {
    for (long long i=p; i<=q; ++i)
    {
      //if (!used[q-p+i]) //проверяем, не использовано ли уже
      {                   //число в выборке
        //used[q-p+i]=1;  (если числа могут повторяться,
                          //не выполнять)
        sett.push_back(i); //вносим число в выборку,
        rek(pr*i);
        sett.pop_back(); //извлекаем число из выборки
        //used[q-p+i]=0;
      }
    }
  }
}

int main() //основная программа
{
  cin>>p>>q>>k; //входные данные
  rek(1); //изначально произвед. =1
  return 0;
}
```

ЛГ-М ≠ ПРОГР

Олимпиада школьников «Надежда энергетики»

МЭИ

Место проведения

ФЫ 31-48

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37101

ФАМИЛИЯ Богданов ~~Сергей~~

ИМЯ Сергей

ОТЧЕСТВО Владимирович

Дата рождения 21.01.2001

Класс: 10

Предмет информатика

Этап: заключительный

Работа выполнена на 6 **листах**

Дата выполнения работы: 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Бог

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37101

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ФЫ 31-48

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
N1  цел B, Q, P, i, l, j;
    лог check, bol;
начало
  ввод (P, Q, B);
  для i от P до Q нц
    bol=true;
    для l от 2 до sqrt(i) нц
      если (i mod l =0) то        //если дел., то провер.
        check=true;               // на простоту
        для j от 2 до sqrt(l) нц  // до его корня
          если (l mod j=0) то
            check=false;
            break;
          всё;
        кц
        если (check) то
          если (l >B) то
            bol = false;          //если больше B,
          всё;                    // то такое „a"
        всё;                      // и не подходит
      всё
      если (not bol) то
        break;
      всё;
    кц;
    если (bol) то                 // ↓ пробел
      вывод (i); вывод (' ');
    всё;
  кц;  // также можно было решать и
конец  // найти все простые числа и оттак.
       // от этого
```

±

N5 – НЕТ

// нужен так

```
N3 цел n, A[n][n], B[n][n], min, i, j, r1, r2, r3,
r4;
начало
  ввод (n);
  для i от 1 до n
    нц
    для j от 1 до n
      нц
      ввод (A[i][j]);
      кц
    кц
  для i от 1 до n
    нц
    для j от 1 до n
      нц
      min = -1;  // присв. min=-1, чтобы ~~1 ому эл. присв.~~
                 // ~~присв.~~ потом сделать min =
                 // 1 элемент треугольника.
      если ( j > i-1) то // ниже диагонали
        r1 = j-i+1;  // до диаг
        r2 = 0; r4=
        пока (r2 ≤ r1)
          нц
          для r3 от i до i+r2+1
            нц
            если (min=-1) то //если 1 вхожд.
              min= A[i+r2][r3];
            всё;
            иначе
            если (A[i+r2][r3] < min)
```

```
min = A[i+r2][r3];
все;
все;
кц;
r2 = r2+1;
кц
B[j][i] = min;
все;
иначе // выше диагонали
~~r1 = abs(j-i+1);~~
r1 = abs(j-i+1); // расч-ие до диаг.
r2 = r1; r4=0;
пока (r2 > 0)
нц
для r3 от r1 до r2 шаг -1
нц
если (min==-1) то // 1-е вхождение
min = A[j-r4][r3];
все;
иначе
если (A[j-r4][r3] < min) то
min = A[j-r4][r3];
все;
все
кц
r2 = r2-1; r4 = r4-1;
кц
B[j][i] = min;
все;
```

```
    кц
  для i от 1 до n        //вывод
   нц
   для j от 1 до n        // ↓ пробел
     нц
     вывод (B[i][j], ' ');
     кц  // ↓перевод строки
   вывод ('/n');
   кц
конец
```

№2 -

```
цел i, a, fin, mas[256], p, k, S1, S2;
   лог check, f1, f2;
начало
 для a от 1 до 256       // цикл - число
  нц check=true;
  для i от 1 до 256       // цикл - степень
   нц
   p=pow(a,i); //pow - возведение в степень i
                              // числа a
   p=p mod 257;
   ~~mas~~
   для k от 1 до i-1   // цикл - сравнение
    нц                  и на сходство
    если (mas[k]=p) то
      check=false;      // если сходство найдено
      break;
      все;
    кц
   если (not check) то
    break;
    все; иначе  mas[k]=p; все;
   кц
```

```
если (check) то
  f1=false;
  f2=false;
  для k от 1 до 256 //цикл-поиск
    нц
    если (mas[k]=9) то
      f1=true; s1=k;
    все;
    если (mas[k]=256) то
      f2=true; s2=k;
    все;
    кц
  если (f2)and(f1) то //если оба найдены
    p=pow(a, s1*s2);
    вывод (p mod 257);
    вывод (' '); //пробел
  все;
  все
кц
конец
```

(+)

№4 ~~цел n, first [2][], second [2][]~~
цел n, first [2][n], second [2][n], ~~tmp~~, i;

начало
ввод (n);
для i от 1 до n
нц
ввод (first [1][i]);
ввод (first [2][i]);
кц
для i от 1 до n-1
нц
second [2][i+1] = first [2][i];
кц
second [1][n] = first [2][n];
для i от n до 2 шаг -1
нц
second [1][i-1] = first [1][i];
кц
second [2][1] = first [1][1];
для i от 1 до n //↓ пробел
нц
вывод (second [1][i], ' ', second [2][i]);
кц
//вывод через пробел
конец
//↓ перевод строки
вывод ('/n');

Олимпиада школьников «Надежда энергетики»

МЭИ, МОСКВА

Место проведения

ЙR86-73

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 3711

ФАМИЛИЯ ВАСИЛЬЕВ

ИМЯ ВАЛЕРИЙ

ОТЧЕСТВО МАКСИМОВИЧ

Дата рождения 09.06.1999

Класс: 11

Предмет ИНФОРМАТИКА

Этап: ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 7 **листах**

Дата выполнения работы: 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Каѕу

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№1

Комментарий

```
АЛГ Проверка():
НАЧАЛО
ЦЕЛ B, P, Q
ВВОД B, P, Q
НЦ
МАС del[Q-P] = []
ЦЕЛ K
ДЛЯ K от P до Q:
   НЦ
   ЦЕЛ A
   ДЛЯ A от 2 до K-1:
   НЦ
   ЕСЛИ K(mod A) == 0:
       K+=1
       КЦ
   КОНЕЦ ЕСЛИ
   del ДОБАВИТЬ K
   КЦ
   КЦ  del ДОБАВИТЬ Q+1
ЦЕЛ i
ДЛЯ i от P до Q:
   НЦ
   ЦЕЛ n
   ДЛЯ ВСЕХ n ИЗ del:
      НЦ
      ЕСЛИ n < i:
         ЕСЛИ i(mod n) == 0:
            ЕСЛИ n >= B:
               ВЫВОД "ЧИСЛО", i, "НЕ", B, "-ГЛАДКОЕ"
               i+=1
            КОНЕЦ ЕСЛИ
         КОНЕЦ ЕСЛИ ↓
```

Ввод

массив простых делителей от P до Q

находим все простые числа от P до Q, добавляем в массив

– чтобы выполнялось если для числа Q

для всех чисел от P до Q проверяем все ли их простые делители меньше B

Олимпиада школьников «Надежда энергетики»

ВАРИАНТ: 37171 ШИФР НЕ ЗАПОЛНЯТЬ! ⇨ ЙR86-73

N1 (продолжение)

ИНАЧЕ:

ВЫВОД "ЧИСЛО", i, " ", B, "-ГЛАДКОЕ"

КЦ

КЦ

КОНЕЦ

N2

Известно, что при умножении числа в степени a на это же число в степени b, то степени складываются.

То есть если $m^a \pmod k = i$, а $m^b \pmod k = j$, то $m^{a+b} \pmod k = i + j$

К решению не относится //

N2

Переберем все натуральные числа a_k от 1 до 256, такие, что $a^{256} \pmod{257} == 1$ и существуют как минимум одно число x, такое, что $a^x \pmod{257} == 9$, и одно число y, такое, что $a^y \pmod{257} == 256$, $1 \le x, y \le 256$. И для всех i различны $R_{257}(a^i)$. Добавим их в массив.

~~Заведем для~~ Пусть для каждого a_k есть массивы xs_k и ys_k, в которых лежат возможные x и y соответственно, а ~~k~~ a_k - все ~~числа~~ возможные числа, для которых выполняется первое условие.

Для каждого a_k посчитаем все возможные произведения xy и вычислим $a_k^{xy} \pmod{257}$.

алгоритм

№2 (продолжение)

```
АЛГ R257 ():
НАЧАЛО
МАС as = []
ЦЕЛ a = 1
ПОКА a < 257:
    НЦ
    МАС xs = [], mas ys = [], mas ost = []
    ЕСЛИ a^256 (mod 257) != 1:
        a += 1
    ИНАЧЕ:
        ЦЕЛ i = 1
        ПОКА i < 257:
        НЦ
          ЕСЛИ a^i (mod 257) в ost:   // если остаток уже в массиве
               a += 1
          ИНАЧЕ:
             ost добавить a^i (mod 257)
          ВСЁ
          ЕСЛИ a^i (mod 257) == 9:
             xs добавить a^i (mod 257)
          ВСЁ
          ЕСЛИ a^i (mod 257) == 256:
             ys добавить a^i (mod 257)
          ВСЁ
          КЦ
       ЕСЛИ xs и ys:   // существуют и x и y
          ДЛЯ ВСЕХ x в xs:
             НЦ
             ДЛЯ ВСЕХ y в ys:
             НЦ
                ВЫВОД "возможное число", a^(x·y) (mod 257)
                КЦ
             КЦ
       ВСЁ
    ВСЁ a += 1
    КЦ
КОНЕЦ
```

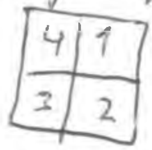
№2 (продолжение)

Если считать, что мы также знаем a, а не только $R_{257}(a^x)$ и $R_{257}(a^y)$, то алгоритм упрощается и не нужно перебирать все a.

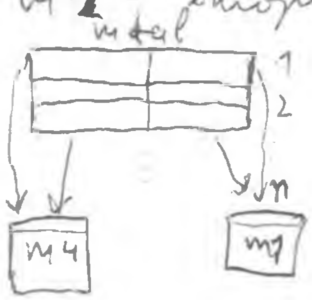
№4

Пусть таблица изначально задана двумерным массивом mtab длиной $2n$ (в нем лежит $2n$ массивов, в каждом из которых лежит одна строчка таблицы в порядке возрастания)

Создадим новые 4 двумерных массива для всех 4 квадратов



Для этого для всех массивов в исходном от 1 до n будем добавлять в массив m_4 первую половину строчки (массива), а в m_1 – вторую половину. А от $n+1$ до $2n$ в m_3 и m_2 соотв.



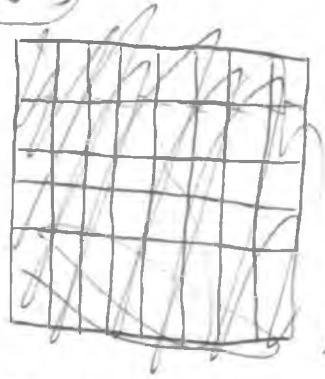
Теперь создадим новый ~~массив~~ ~~с конц~~ двумерный массив, длиной $2n$ – mres

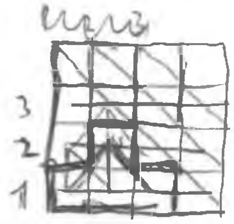
В этот массив будем добавлять сумму двух массивов для от 1 до n из массивов m_3 и m_4 и от $n+1$ до $2n$ для массивов m_2 и m_1

В результате получим двумерный массив, в котором квадраты поменяны местами как на рисунке 2.

Каждый i-тый массив задает i-ую строку таблицы

№3





Заменим число элементы в B ~~на диагонали~~ в последней строке оставшиеся равные элементам ~~на главной диагонали~~ в последней строке A.

А элементы на ~~диагонали~~ строке 2 определяются, как минимум из 4 элементов (сам элемент на ~~диагонали~~ строке 2 A и 3 элемента на ~~диагонали~~ строке 1, соседние к нему). Аналогично происходит и с ~~диагональю~~ строкой 3, где элемент - минимум из самого элемента и 3 элементов на ~~диагонали~~ строке 2, соседних.

Пусть матрица A задается двумерным массивом, где каждый i-ый массив определяет ~~i-ую~~ строку матрицы A.

$A = [d_1[k], d_2[k] \dots d_n[k]]$

$d_i[k] = [\underbrace{10^{100}, a, b, c \dots x, 10^{100}}_{n+2 = k \text{ чисел}}]$

Чтобы не было проблем с элементами на сторонах, добавим в каждую строку очень большое число, большее всех в матрице, чтобы около было мин (слева и справа)

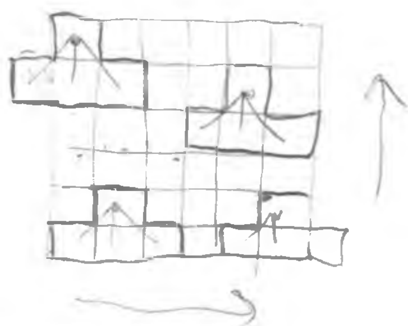
Аналогичный массив заведем для B

```
АЛГ МАТРИЦА():
НАЧАЛО
A = [d1[n], d2[n] ... dn[n]]
B = [t1[n], t2[n] ... tn[n]]
ЦЕЛ i = 2
ПОКА i ≤ n+1:
   НЦ
   B[n][i] = A[n][i]      // добавляем в B элем. последней строки A
   i += 1
   КЦ
```

↓

№3 (продолжение)

```
ЦЕЛ J = n - 1
ПОКА J > 0
  НЦ
    ЦЕЛ i = 2
    ПОКА i ≤ n+1
      НЦ
        B[J][i] = min(A[J][i], A[J+1][i-1], A[J+1][i],
                                      A[J+1][i+1])
      // B[J][i] - минимальное из 4 значений
        i += 1      // проходим по каждому элементу
      КЦ            // строки слева направо для
    J += 1          // строк снизу вверх и задаем
  КЦ                // матрицу B
ЦЕЛ p = 1
ПОКА p ≤ n
  НЦ
    ВЫВОД  B[p]   // вывод получившейся
    p += 1        // матрицы B по строкам
  КЦ              // (каждый массив B[p] -
КОНЕЦ             // p-строка в матрице)
```



№5

Поскольку оно не равно произв. мн-е 1, то длина множества больше 2.
Так же в этом множестве ~~всегда уголовна~~ может быть одно четное число, но не более.

№5 (продолжение)

Иначе для этих четных чисел произведение +1 будет нечетным.

~~Если данная ... не будет, то произведение~~

```
АЛГ ПРОВЕРКА (мас m):
НАЧАЛО
ЦЕЛ i = 1
ЦЕЛ r = 1
ПОКА i ≤ длина (m):
   НЦ
   ЦЕЛ j = 1
   ЦЕЛ p = 1
   ПОКА j ≤ длина (m):
      НЦ
      ЕСЛИ j != i
           p = m[j]·p
      ВСЁ
      j += 1
      КЦ
   p += 1
   ЕСЛИ (p mod m[i]) != 0:
      ~~ВЫВОД «МАССИВ НЕ ПОДХОДИТ»~~ r = 0
      i = длина (m)
   ИНАЧЕ
      i += 1
   ВСЁ
ЕСЛИ r == 1
   ВЫВОД 1
ИНАЧЕ
   ВЫВОД 0
ВСЁ
КОНЕЦ
```

Запустим алгоритм

Проверка со всеми возможными перестановками целых нечетных чисел от -10^{15} до 10^{15}, длиной k.

И со всеми перестановками аналогично, длиной $k-1$ и еще четным числом от -10^{15} до 10^{15}.

Сумма выводов и будет результатом для k

алгоритм не эффективен по времени, но в итоге результат находит

Олимпиада школьников «Надежда энергетики»

МЭИ, Москва

Место проведения

ЙR 86-80

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ Гладкова

ИМЯ Анастасия

ОТЧЕСТВО Александровна

Дата рождения 02.12.1999

Класс: 11

Предмет Информатика

Этап: Заключительной

Работа выполнена на 6 **листах**

Дата выполнения работы: 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Гладкова

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Задача № 1.

Нам нужно разработать алгоритм, который проверяет, является ли число в диапазоне от P до Q B-гладким, т.е нам нужно проверить, все ли его простые делители не превосходят заданного числа B. Для начала нам нужно ввести само число B, а затем организовать цикл от P до Q. Через этот цикл мы начнем проверять каждое число от P до Q. Начнем для каждого числа от P до Q искать простые делители через цикл for i:=2 (т.к 1 - не простое число) to (P div 2) do. Если i является делителем числа P, проверяем, является ли оно простым (то есть делится только на себя и на 1) через цикл for j:=1 to i do. Если (i mod j =0) то увеличиваем счетчик k на 1. В конце цикла k обнуляем если число простое, k будет равняться 2. Если это так и число i простое, то присваиваем переменной k1 значение k1:=k1+1, и дальше проверяем, не превосходит ли число i числа B, если нет, то присваиваем переменной k2 значение k2:=k2+1. Если число является B-гладким, то значения k2 и k1 на выходе из циклов совпадут, и мы выведем („число", P, "B-гладкое"). Если же k2 не равняется k1, то не все простые делители числа P не превосходят B, поэтому мы выведем („число", P, "не B-гладкое").

+

Задача № 2.

Нам известо, что при подобранном числе a, числа $R_{257}(a^i)$ различна при всех $1 \leq i \leq 256$, то есть

a^i mod 257 = одно число

a^{i+1} mod 257 = другое число

a^{i+2} mod 257 = третье число и т.д

Также нам известно, что:

$R_{257}(a^x) = 9$, т.е $a^x \bmod 257 = 9$

$R_{257}(a^y) = 256$, т.е $a^y \bmod 257 = 256$

$R_{257}(a^{256}) = 1$, т.е $a^{256} \bmod 257 = 1$

Известно, что $a \leq 256$, $x \leq 256$ и $y \leq 256$

Для начала нам нужно найти такие числа a, чтобы выполнялись условия:

1) При всех $1 \leq i < 256$ числа $R_{257}(a^i)$ различны

2) $R_{257}(a^{256}) = 1$

Если таких чисел будет несколько, проверяем каждое из них на ~~выполнения~~ других условий

1) $a^x \bmod 257 = 9$

2) $a^y \bmod 257 = 256$

Т.е при $x \leq 256$ и $y \leq 256$ ищем такие x и y, чтобы выполнялись эти 2 условия. Если при данном a нет таких x и y, переходим к следующему подходящему числу a и проверяем данные 2 условия уже с ним.

Когда будут найдены подходящие числа a, x, y, ищем секретное число $R_{257}(a^{xy})$, то есть $a^{xy} \bmod 257$, для этого возводим число a в степень x, а затем полученное число возводим в степень y и находим остаток от деления этого числа на 257.

Задача №3

Нам дана матрица A размера n, нужно построить матрицу B того же размера, где b_{ij} определяется особым образом.

Возьмем, например, такую матрицу A:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Для нахождения значения b_{ij} через a_{ij} проведем диагонали, параллельные главной и побочной диагоналям, т.е

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Из тех чисел, которые оказались под штриховкой нам нужно выбрать наибольшее число и записать его значение в b_{ij}. Таким числом является 16.

Нетрудно заметить, что числа, которые оказались под штриховкой, начинаются со значения a_{ij}, дальше с каждой строчкой увеличивается i, при этом изменяется значение и j, если это возможно (пока $i \le n$; $j \le n$)

a[1,1]

a[2,1] a[2,2]

a[3,1] a[3,2] a[3,3]

a[4,1] a[4,2] a[4,3] a[4,4]

Дальше не рассматриваем т.к i будет больше n.

i при каждом шаге увеличивается на 1, пока $\le n$, j не может меняться с каждым шагом на несколько значений, т.к как полученная заштрихованная фигура - треугольник, но j обязательно должно быть ≥ 1 и $\le n$.

Для каждого a_{ij} мы найдем наибольшее среди чисел, попавших под штриховку, и запишем их значение в b_{ij}

a[1,1]

a[2,0] a[2,1] a[2,2]

не берем, $j < 1$

Олимпиада школьников «Надежда энергетики»

ВАРИАНТ: 37111 ШИФР НЕ ЗАПОЛНЯТЬ! ⇨ ЙR86-80

a[3,-1] не берем, j<1 a[3,0] не берем, j<1 a[3,1] a[3,2] a[3,3]

a[4,-2] не берем, j<1 a[4,-1] не берем, j<1 a[4,0] не берем, j<1 a[4,1] a[4,2] a[4,3] a[4,4]

С каждым увеличением i у нас уменьшается минимальное значении j и увеличивается максимальное значении j на 1. В этом диапазоне мы и рассматриваем числа, если j≥1 и j≤n.

Задача № 4

Нам нужно получить новую таблицу, переставляя блоки размера n×n, как показано на рисунке. Всего блоков размера n×n будет 4

1	2
3	4

нам нужно поменять местами 1 и 4 блоки, 3 и 2 блоки

Внутри каждого блока может быть несколько чисел, поэтому нам нужно создать 2 цикла:
1) Первый цикл будет менять местами блоки 1 и 4
2) Второй цикл будет менять местами блоки 2 и 3

Алг-м ≠ прогр

Первый цикл:

```
for i:= 1 to n do begin
for i1:=(n+1) to 2*n do begin
for j:= 1 to n do
for j1:= (n+1) to 2n do
a[i,j]= x;
a[i1,j1]=y;
a[i,j]= y;
a[i1,j1]= x;
 end; end;
```

С помощью этого цикла мы поменяем блоки 1 и 4

Второй цикл:

```
for i := (n+1) to 2n do begin
for i1: 1 to n do begin
for j := 1 to n do
for j1 := (n+1) to 2n do
a[i,j] := x;
a[i1,j1] := y;
a[i,j] := y;
a[i1,j1] := x;
end; end;
```

С помощью этого цикла мы поменяем блоки 3 и 2

Например.

	1	2	3	4
1	1	2	5	6
2	3	4	7	8
3	9	10	13	14
4	11	12	15	16

a[1,1] будет равно 13
a[2,1] - 15
a[3,1] - 5
a[4,1] - 7 и тд

В итоге мы получим:

13	14	9	10
15	16	11	12
5	6	1	2
7	8	3	4

Задача №5

Нам нужно найти количество решений задачи Знаша для k в диапазоне от P до Q. Нам нужно рассмотреть каждое множество k целых чисел в диапазоне от P до Q и найти такие множества, где каждое число является делителем произведения других целых чисел в множестве плюс 1.

Начнем по отрезности рассматривать каждое множество, состоящее из k элементов, входящее в диапазон от P до Q. для каждого элемента множества мы проверяем, является ли оно делителем произведения других чисел в множестве + 1. Если элемент является решителем, то увеличиваем переменную k1 на 1. Когда мы проверим все элементы множества и получим значение k1 равное k, то выясним, что это множество является решением задачи Знама. Если множество является решением задачи Знама, то увеличим значение переменной ch на один (ch будет считать кол-во решений задачи Знама). k 1 присвоим значение 0 и начнем проверять следующее подходящее множество. Когда мы проверим все подходящие множества, то посчитаем число решений задачи Знама и выведем их количество (ch)

Дополнительно: ни один элемент подходящих множеств не должен быть равен произведению других элементов этого же множества +1. Если хоть один элемент подходящего множества будет равен произведению других элементов этого же множества +1, то мы сразу переходим к другому множеству и проверяем уже его.

Олимпиада школьников «Надежда энергетики»

Лицей №18

Место проведения

9S83-50

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37091

ФАМИЛИЯ Гнелицкий (ГНЕЛИЦКИЙ)

ИМЯ Вячеслав (ВЯЧЕСЛАВ)

ОТЧЕСТВО Иванович (ИВАНОВИЧ)

Дата рождения 23.11.2000 **Класс:** 9

Предмет Информатика **Этап:** Заключительный

Работа выполнена на 2 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: ГВИ

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37091

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ЯS83-50

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№1
```
def foo(n):
    if sqrt(n) < b:
        return True
    elif sqrt(n) == b:
        return False
    for i in range(2, sqrt(n)+1):
        while n%i == 0:
            if lst[-1] != i:
                lst.append(i)
            n //= i
    for i in lst:
        if i > b
            return False
    return True
for i in range(P, Q+1):
    print(i, foo(i))
```

+

№2 Бинарный поиск
```
l = 1
r = n
while True:
    m = (l+r)//2
    if m*(m+2) > n:
        r = m
    elif m*(m+2) < n:
        l = m
    else:
        return (m, m+2)
```

Нет необходимости искать только среди простых чисел, потому что существует только одно число m такое, что m(m+2) = n

№5.
С каждым ходом черепаха будет проходить все меньшее расстояние, и через некоторое время оно будет меньше наименьшей разницы между числами с плавающей точкой, для компьютера оно будет равно 0, и тогда черепаха остановится, а Ахилл вырвется вперед ~~(и через некоторое время тоже остановится)~~

±

№3. Рассмотрим следующие ситуации:

1) Если все числа в ~~последовательности~~ множестве нечётные, то П+1 (произведение остальных + единица) — чётное, и ~~оно~~ ~~при~~ условие выполняется только в последовательностях, полностью состоящих из единиц.

2) Если все числа чётные*, то П+1 — нечетное, и оно не разделится ни на одно из чётных чисел

* достаточно двух чётных чисел в множестве, могут и присутствовать нечётные

(−)

3) если одно из чисел четное, то условие ~~не~~ будет выполняться ~~[2,1,1,1]~~ только в последовательностях, состоящих из единиц и одной двойки**, но тогда $2 = 1 \cdot 1 \cdot \ldots \cdot 1 + 1$, что тоже не удовлетворяет условию

Значит, единственные множества, удовлетворяющие условию, состоят из k единиц, для любого k существует только одно решение

** не обязательно, например: $\{1, 2, 3\}$, но здесь $3 = 2 \cdot 1 + 1$

№4. Найдем простые числа от 2 до q решетом Эратосфена.

```
numbers = [True for i in range(0, q+1)]
numbers[0] = numbers[1] = False
primes = [ ]  # список для хранения простых чисел
per = dict(int) # Словарь в языке python
for i in range(2, q+1)
    if numbers[i]==True
        for j in range(i**2, q+1, i).
            numbers[j] = False
        if u≤i≤w:   (зачёркнуто)
            per[foo(i)] += 1  # foo(i) считает длину
                              # периода числа i
        if u≤i≤w:
            primes.append(i)
for i in primes
    if per[foo(i)]>1:
        print(i, "Не является уникальным")
    else:
        print(i, "Является уникальным")
```

Алг-м ≠ прогр

КАК?

±

Олимпиада школьников «Надежда энергетики»

МЭИ

Место проведения

ЙR86-42

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ ДЖИДЖОЕВ

ИМЯ ВЛАДИСЛАВ

ОТЧЕСТВО Муратович

Дата рождения 07.10.1999

Класс: 11

Предмет информатика

Этап: заключительный

Работа выполнена на 5 **листах**

Дата выполнения работы: 18.02.2017г
(число, месяц, год)

Подпись участника олимпиады:

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 27111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ЙR86-42

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

Задание 1.

Для решения этой задачи модифицируем алгоритм "Решето Эратосфена": для каждого числа от до Q будем хранить 0, если оно простое, либо его максимальный простой делитель. Простой делитель поддерживает максимальным, если по ходу работы алгоритма для каждого числа, которое делится на i записываем i как макс. делитель (т.к. мы перебираем i от меньшего к большему, делитель i будет максимальным для текущей итерации алгоритма).

(+)

Алгоритм:

1. Ввести P, Q, B
2. Создать массив A размером Q ~~и заполнить его нулями~~.
3. Для каждого i от 1 до Q: ~~Если~~ Если P = 1, то вывести „1 - B-гладкое".
3.1. Записать в A[i] 0.
4. Для каждого i от 2 до Q.
4.1. Если A[i] ≠ 0:
4.1.1. Если A[i] ≤ B и i ≥ P, вывести i, „B-гладкое число"
4.1.2. Иначе если A[i] > B и i ≥ P, вывести i, „не B-гладкое число"
4.2. Иначе:
4.2.1. Если i ≤ B и i ≥ P, вывести i, „B-гладкое число".
4.2.2. Для каждого j от от 2·i до Q с шагом i:
4.2.2.1. Записать в A[j] i.
4.2.3. Если i > B и i ≥ P, вывести i, „не B-гладкое число".

Примеч.: „для каждого j от 2·i до Q с шагом i" означает перебор всех j, таких, что j > i, j ⋮ i, j ≤ Q.

Задание 3.

Для эффективного решения этой задачи отметим, что для эффективного решения этой задачи отметим, что

Для эффективного решения этой задачи отметим, что можно составить, отдельные области, закрашенную на рис. 1 область можно составить, и области, максимум которых занятую клеткой a_{ij}, и области, максимум которых хранится в $b_{i,j+1}$, $b_{i+1,j+1}$, $b_{i-1,j+1}$ (будем индексировать матрицы так: a_{ij} это число в j строке и i столбце матрицы a, то же и для b). Тогда очевидно, что b_{ij} определяется как $b_{ij} = a_{ij}$, для $j = n$

$b_{ij} = \max(a_{ij}, b_{i,j+1}, b_{i+1,j+1}, b_{i-1,j+1})$, для всех $j \neq n$ и i - внутри матрицы

Лист 01 из 05

Получается, можно эффективно посчитать B_{ij} за время $O(n^2)$ снизу вверх (т.к. B_{ij} может зависеть только от значений матрицы в строке $j+1$).

Алгоритм:

1. Ввести n. Создать матрицу B размером n×n.
2. Ввести матрицу A размером n на n.
3. Для каждого i от 1 до n: ← (заполняем нижнюю строку)
3.1. Приравнять $B_{i,n}$ к $A_{i,n}$.
4. Для каждого j ~~от n-1 до 1~~ от 1 до n-1 (перебор от большего к меньшему): ← (строим B)
4.1. Для каждого i от 1 до n:
4.1.1. $B_{ij} := A_{ij}$ (приравнять B_{ij} к A_{ij})
4.1.2. B_{ij} приравнять к максимуму из B_{ij} и B_{ij+1}
4.1.3. Если $i>1$:
4.1.3.1. B_{ij} приравнять к максимуму из B_{ij} и $B_{i-1;j+1}$
4.1.4. Если $i<n$
4.1.4.1. B_{ij} приравнять к максимуму из B_{ij} и $B_{i+1;j+1}$
5. Вывести B на экран.

Задача 4.

Алгоритм:

1. Ввести n.
2. Ввести матрицу A размером 2n × 2n.
~~3. Создать матрицу B размером 2n×2n~~
3. Для каждого j от 1 до n: ← (меняем местами левый верхний и правый нижний блоки)
3. Для каждого i от 1 до n:
3.1. Для каждого i от 1 до n:
3.1.1. Приравнять t к A_{ij}
3.1.2. Приравнять A_{ij} к $A_{i+n;j+n}$ ← меняем местами A_{ij} и $A_{i+n;j+n}$
3.1.3. Приравнять $A_{i+n;j+n}$ к t.
4. Для каждого j от n+1 до 2n: ← (меняем местами правый верхний с левым нижним блок)
4.1. Для каждого i от 1 до n:
4.1.1. Приравнять t к A_{ij}
4.1.2. Приравнять A_{ij} к $A_{i+n;j-n}$
4.1.3. Приравнять $A_{i+n;j-n}$ к t
5. Вывести A.

Пояснение: Просто от ячейки к ячейке меняем блоки. Нумерация: A_{ij} – i-ый столбец, j-ая строка A.

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

Очевидно, что сложност алгоритма $O(n^2)$.

Задание 2.

~~[illegible]~~

$a^{xy} \equiv (a^x)^y \equiv (a^y)^x \pmod{257}$ (из св-в остатков и степеней).

~~Следовательно, чтобы найти ... надо $R_{257}(a^y)$ ...~~

Все числа a, x, y не превосходят 256 ⇒ эту задачу можно достаточно эффективно решить перебором: сложност алгоритма $O(n^3)$, где (общ кол-во итерацийт $\leq 10^9$; перебор на компьютере займет не более секунды). $n = 256$, при таком n перебор ... все $R_{257}(a^i)$ различны, заведем Чтобы проверят эффективно, что массив ~~R_{ij}, в котором R_{ij} элемент будет хранить количество~~ массив R_{ij} 257 элементов, в котором в R_j будет хранится количество $R_{257}(a^i)$, равных $j-1$, при фиксированном a и перебираемом i. Нумерацию всех массивов ведем с 0 до $n-1$.

~~Алгоритм:~~

~~1. Перебираем a от 1 до 256:~~

~~1.1. Создаем заполненный нулями массив R из 257 элементов.~~

~~1.2. Пусть $b := 1$ ← (будем перебирать степени a^i на каждое итерации цикла домножая предыдущую степень на a)~~

~~1.3. Перебираем i от 1 до 256:~~

~~1.3.1. b приравняем к $b \cdot a$ ($b := b \cdot a$)~~

~~1.3.2. b приравняем к остаку от деления b на 257.~~

~~1.3.3. Если $R[b+1] = 1$: ← (проверка на различность остатков)~~

~~1.3.3.1. Перейти к шагу 1.1 (следующая итерация, текущее a не подходит)~~

~~1.3.4. Приравнять $R[b+1] := R[b+1] + 1$~~

~~1.4. Если $b \neq 1$: ← (проверка $R_{257}(a^{256}) = 1$)~~

~~1.4.1. Перейти к след. итерации цикла (шаг 1.1)~~

1\. Перебираем все a от 1 до 256:

1.1. Создаем заполненные 0 массивы R и B из 257 эл-тов (R_j – количество a^i, дающих остаток $j-1$; B_j – остаток a^j при делении на 257)

1.2. Пусть $B_0 := 1$

1.3. Последовательно перебираем все i от 1 до 256:

1.3.1. Пусть $B_i = B_{i-1} \cdot a \bmod 257$ (получаем $a^i = a^{i-1} \cdot a$, $x \bmod 257$ – остаток от деления x на 257)

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ЙR86-42

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

1.3.2. Если $R[B_i \cdot N] = 1$: (проверка кол-ва раз остатков, они не должны превосходить)

1.3.2.1. Перейти к след. итерации цикла 1 (шаг 1.1)

1.3.3. $R[B_i \cdot N] := R[B_i \cdot N] + 1$ ← (а не подходит, т.к. дает несколько раз один остаток)

1.4. Если $B_{256} \neq 1$:

1.4.1. Перейти к след. итерации цикла 1 ($R_{257}(a^{256}) \neq 1$)

1.5. Перебираем все x от 1 до 256:

~~1.5.1. Перебираем все y от 1 до 256~~

~~1.5.1.1. Если $B_x = 9$ и $B_y =$~~

1.5.1. Если $B_x = 9$:

1.5.1.1. Перебираем все y от 1 до 256:

1.5.1.1.1. Если $B_y = 256$: ← ($R_{257}(a^x) = 9$ и $R_{257}(a^y) = 256$, искомые x, y, а найдены)

1.5.1.1.1.1. Вывести ответ $B_{x \cdot y \bmod 256}$ ← (нам не надо считать a^{xy}, т.к. $a^{256} = 1 \Rightarrow$ $\Rightarrow a^{xy} = a^{xy \bmod 256}$)

1.5.1.1.1.2. Завершить алгоритм

1.6. Вывести „ответ не найден" (если алгоритм продолжился после цикла, значит ответ не найден)

Задание 5.

Рассмотрим случай двухэлементных множеств $\{n_1, n_2\}$. подходит тогда, когда $n_1 | (n_2+1)$ („|" – „делит"), а $n_2 | (n_1+1)$. ~~Очевидно, это подходят только мн-ва $\{1;1\}$, $\{1;2\}$, ... т.е. для $n=2$ ответ 3.~~

Заметим, что если в мн-ве чётное кол-во элементов n, то при любом n-ом a n−1 элемент равен −1, то при любом n-ом элементе $a_n \in N$ мн-во $\{-1; -1; \ldots; a_n\}$ будет решением задачи. Если кол-во элементов нечётное, то решением мн-ва $\{-1; \ldots; -1; 1; x\}$ $\forall x \in N$ будут решения $\Rightarrow$ ~~$\forall k$~~ $\forall k \geq 3$ существует бесконечное мн-во решений.

Алгоритм:

1. Считать P и Q

2. Если $Q \geq 3$ вывести ответ: „~~бесконечно~~ $10^{17}+1$ ~~много~~"

3. Если $Q = 2$ вывести „~~бесконечно~~ $10^{17}+1$ ~~много~~"
(подходят все $\{-1; x\}$ $x \in \mathbb{N}$ и $\{1; 1\}$)

~~4. Если $Q = 1$~~

Олимпиада школьников «Надежда энергетики»

ГОРОД КРАСНОЯРСК
Место проведения

0109UK
шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37091

ФАМИЛИЯ КАЗАКОВ

ИМЯ МИХАИЛ

ОТЧЕСТВО ВЯЧЕСЛАВОВИЧ

Дата рождения 21.07.01 **Класс:** 9

Предмет ИНФОРМАТИКА **Этап:** ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 6 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Каз

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

№1
Для начала найдем все простые числа в диапазоне от 2 до $\sqrt{Q}$.
Далее будем проверять делимость каждого числа в диапазоне от P до Q на все числа j из диапазона простых чисел от 2 до $\sqrt{Q}$ (при условии, что $j \leq i$).
Код ниже написан на языке Python.

```
~~fromm~~ from math import sqrt
B= int(input())
P= int(input())
Q= int(input()); Primes=[]; ans=[]
for i in range(2, ~~sqrt(Q)):~~ sqrt(Q)+1):
    for j in range(2, sqrt(i)+1):
        if i % j==0:
            break
    else:
        primes.append(i)
primes.append(2) # так как в результате алгоритма
# выше 2 не добавляется в primes
for i in range(P, Q+1):
    for j in primes:
        if j>i:
            break
        elif i % j==0:
            if j>=B:
                ans.append(i)
            break
```

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
# Усилим алгоритм. Проверим простые числа
# в диапазоне от B+1 до Q.
for i in range(B+1; Q+1):
    for j in range(2; sqrt(i)+0,001):
        if i % j == 0:
            break
    ~~else:~~
    else:
        ans.append(i)
for i in range(P; Q+1):
    if i in ans:
        print(i, 'не является', B, '-гладким')
    else:
        print(i, '-', B, '-гладкое')
```

№2

Заметим, что уравнения симметричны, так как если поменять p и q местами, то вид не изменится:

$p \cdot q = q \cdot p$;

$|p-q| = |q-p|$.

Преобразуем уравнения.

$p-q=2$

$p=2+q$; $n=p \cdot q$

$\frac{p \cdot q}{p} = \frac{n}{2+q}$; $q(q+2)=n$ или $p(p+2)=n$

$q^2+2q-n=0$ или $p^2+2p-n=0$

$q=-1\pm\sqrt{1+n}$ $\quad$ $p=-1\pm\sqrt{1+n}$

$q=-1\pm\sqrt{400032000064}$ $\quad$ $p=-1\pm\sqrt{400032000064}$

$n=p(-1\pm\sqrt{400032000064})$ $\quad$ $n=q(-1\pm\sqrt{400032000064})$

$p=\dfrac{400032000063}{-1\pm\sqrt{400032000064}}$ $\quad$ $q=\dfrac{400032000063}{-1\pm\sqrt{400032000064}}$

Посчитаем значения выражений.

Язык программирования: Python.

```
from math import sqrt
~~print('q=', -1+sqrt(400032000064), 'p=', 400032000063/(-1+sqrt(400032000064)))~~
n=400032000063
chisla=[(-1+sqrt(n+1), n/(-1+sqrt(n+1))),
        (-1-sqrt(n+1), n/(-1-sqrt(n+1))),
        (n/(-1+sqrt(n+1)), -1+sqrt(n+1)),
        (n/(-1-sqrt(n+1)), -1-sqrt(n+1))]
# Проверяем, какие числа простые.
for i in chisla:
    for j in i:
        for g in range(2, sqrt(j)+0,001):
            if j % g == 0:
                break
        else:
            continue
        break
```

```
else:
    print('q=', i[0], ';p=', i[1])
```

№5

Я считаю, что программа также не сможет определить результат, так как точный результат можно получить лишь из уравнения:

$$\frac{S+100}{10v} = \frac{S}{v}$$, где v - скорость черепахи, S - её путь.

$S+100 = 10S$

$9S = 100$

$S = \frac{100}{9}$

$S = \cancel{11,(09)}$ $11,(1)$

Число S бесконечно, что подтверждает невозможность его точного определения.

Ниже приведен алгоритм моделирования ситуации на языке Python:

```
k=100
while k!=0:
    k/=10
print('Ахилл догнал черепаху')
```

№4

Алг-м + прог

Найдем все простые числа q в диапазоне от 2 до 10^9. Далее найдем такие числа P, что $U \leq P \leq W$, $P \neq \{2;5\}$.

Посчитаем длину периода для каждого $\frac{1}{P}$ и $\frac{1}{q}$, зная, что $0,(3) = \frac{3}{9}$; $0,(45) = \frac{45}{99}$; $0,0(4) = \frac{4}{90}$ и сравним результаты.

Программа ниже написана на языке Python

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
from math import sqrt
U= int(input())
W= int(input())
q=[]
q_per=[]
p=[]
p_per=[]
for i in range(2,10**9):
    for j in range(2, sqrt(i)+0,001):
        if i%j==0:
            break
    else:
        if i==2 or i==5:
            continue
        else:
            q.append(i)
            if U<= i<=W:
                p.append(i)
for i in q:
    k=9
    while k%i!=0:
        if str(k).count('9')>1:
            k=int(str(k)[:str(k).rfind('9')]+('0'*(str(k)...
.count('0')+1)))
        else:
            k= ~~str~~ int((len(str(k))+1)*'9')
    q_per.append(str(k).count('9'))
```

±

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
if i in p:
    p_per.append(str(k).count('9'))
for i in range(len(p_per)):
    if q_per.count(p_per[i])==1:
        print(p[i], '- уникальное простое число')
    else:
        print(p[i], '- неуникальное простое число')
```

N3

Предлагаю сгенерировать все возможные варианты множества $\{n_1; \dots; n_k\}$, где $n_i < 10^6$, где $k \in [P; Q]$, и сразу проверять удовлетворяет ли множество условию, то есть для любого i n_i делит, но не равно $(\prod_{j \neq i}^{k} n_j + 1)$. Если множество удовлетворяет условию, то к переменной a нужно прибавлять 1, иначе генерировать следующее множество.

Также можно заметить, что при любом k множество $\{1; 1; \dots; 1\}$ удовлетворяет условию.

Олимпиада школьников «Надежда энергетики»

Место проведения

ФН31-21

← **Не заполнять**
Заполняется ответственным работником

шифр

Вариант № 37101

ФАМИЛИЯ Киселев

ИМЯ Алексей

ОТЧЕСТВО Андреевич

Дата рождения 26.12.2000 **Класс:** 10

Предмет ИНФОРМАТИКА **Этап:** ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 4 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Киселев

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37101

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ФН 31-21

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№4

Сначала таблица выглядела так:

1	2
4	3

Потом так

4	1
3	2

Т.е. надо найти отдельно каждый блок и записать его в двумерный массив, а потом построчно заполнить массив таблицы новыми значениями

Псевдокод:

```
a : массив [1..2n, 1..2n] целых чисел
n: целое.
block1, block2, block3, block4 : массив [1..n, 1..n] целых чисел
начало
    ввод n;
    для i:=1 до 2n
      для j:=1 до 2n  ввод a[i,j];        } ввод данных
    для i:=1 до n
      для j:=1 до n нц block1[i,j]:= a[i,j];
         block2[i,j]:= a[i+n, j];
         block3[i,j]:= a[i+n, j+n];          } заполнение блоков
         block4[i,j]:= a[i, j+n];
    конец цикла
    для i:=1 до n
      для j:=1 до n нц a[i,j]:= block4[i,j];
    ~~для i:=n до 2n~~
      ~~для j:=1 до n   a[i,j]:= block1[i-n,j];~~   } заполнение массива блоками
    ~~для~~ a[i+n, j]:= block1[i,j];
        a[i, j+n]:= block3[i,j];
        a[i+n, j+n]:= block2[i,j];
    конец цикла.
конец.
```

N5-НЕТ

№3

Треугольник, в котором находятся варианты b будет

Олимпиада школьников «Надежда энергетики»

Вариант: 37101

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ФН 31-21

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

лежать выше главной диагонали при $i>j$, ниже при $i<j$, и $b_{ij}=a_{ij}$ при $i=j$.

Главная диагональ выражается функцией $x=y$, линии до нее от a_{ij} функциями $x=i$ и $y=j$, из этого следует, что пересечения с главной диагональю будут также в точке с $i=j$.

Псевдокод

```
a, b : массив [1..n, 1..n] целых чисел.
n, min : целое.
  ввод n;
  для i:=1 до n
    для j:=1 до n ввод a[i,j]; }  ввод данных
  для i:=1 до n
    для j:=1 до n нц  min:=maxint //максимальное целое число.
      если i=j то b[i,j]:=a[i,j];
→ иначе если i>j то начало
        для k:=j до i
          для w:=j до i
            если (i>j) и (a[i,j]<min) то min:=a[i,j];
            b[i,j]:=min; конец
→ иначе если i<j то начало
        для k:=i до j
          для w:=i до j
            если (j<i) и (a[i,j]<min) то min:=a[i,j];
      ~~b[i,j]:=min~~
      ~~если b[i,j]<>a[i,j]~~
      b[i,j]:=min;
      конец
```

±

№1

```
Алг гладкость (арг цел P, Q, B)
  начало простые(Q);  z : массив целых чисел.
  для i=P до Q нц
    a:=i;  j:=1;
    пока a≠1 нц
      если a mod z[j]=0 то a:=a div z[j]
      иначе j:=j+1;
```

±

Лист 02 из 04

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
если z[j] > B то начало вывод ('число', i, ' не B-гладкое');
 break; конец;
кц
если a = 1 то вывод ('число', i, ' B-гладкое');
кц
конец

Алг простые (арг цел Q); // алгоритм нахождение простых
 начало                     чисел от 1 до Q
   z[1] := 2;
   z[2] := 3;  k := 3;
   для i := 5 до Q нц
     если (i mod 2<>0) и (i mod 3<>0) то начало
       для j := 3 до i-1
         если i mod z[j] = 0 то goto 1;
       z[k] := i;
       k := k+1;
     конец.
   1: кц.
 конец
```

№2

$R_{257}(a^x)=9$ можно представить как $a^x = 9+257n$

$R_{257}(a^y)=256 \to a^y = 256+257m$

$R_{257}(a^{256})=1 \to a^{256} = 257z+1$

Из трех уравнений выражаем

$$\frac{a^x-9}{n} = \frac{a^y-256}{m} = \frac{a^{256}-1}{z}$$

При этом $a_{max} = 256$, $a^{xy}_{макс} = 256^{256^2}$

$a^{xy} = (9+257n)^y$.

Псевдокод.

~~a, x, y, m, n, z : целое // алгоритм целое типа int64 или long long, т.е. максимально~~

```
начало
для i := 1 до 256
  для j := 1 до 250
    для k := 1 до 256
      для l := 1 до 256
```

для $w := 1$ до 256

для $e := 1$ до 256

если $\frac{i^j - 9}{k} = \frac{i^t - 256}{w} = \frac{i^{256} - 1}{e}$ то

~~начало~~

~~$a := i$;~~ вывод $(9 + 257k)^t \bmod 257$.

~~$x := j$;~~

~~$t := k$;~~

~~$y := t$;~~

конец.

$\frac{?}{+}$

Олимпиада школьников «Надежда энергетики»

МЭИ

Место проведения

ЙR86-93

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ Королев

ИМЯ Дмитрий

ОТЧЕСТВО Павлович

Дата рождения 24.10.1999 **Класс:** 11

Предмет информатика **Этап:** очный, финальный

Работа выполнена на 06 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады:

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37|11 ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ИR86-93

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№1

Для такого алгоритма следует использовать алгоритм „решето Эратосфена". Этот алгоритм позволяет найти простые числа в заданном промежутке. Затем, имея массив простых чисел, которые могут быть делителями данных чисел от P до Q, можно легко проверить простые делители чисел меньшие или равные B?

Такое решение тратит время на построение решета Эратосфена (и память), но затем легче выполнять итерации с числами от P до Q.

~~Программа~~: В условии ~~указано~~ не указано максимальное принимаемое значение Q. пусть Qнаибольшее = 100000. соответственно Bнаиб = 1000000.

Программа:

```
конст Bmax=1000000; Qmax=100000;
цел B, P, Q; ВВОД B, P, Q; цел mass[1000]; //на массив простых чисел mass
                                            //можно создать память динами-
                                            //чески для меньших затрат
                                            //памяти. В C++: „*mass[];"
если ((B<=0) или (B>Bmax)) то
    вывод "ошибка. Введеные значения неверны";
    конец программ, return 0;
конец
если ((P<=0) или (P>Q)) то
    вывод "ошибка. Введеные значения неверны";
    конец программы. return 0;
кон.
если ((Q<=0) или (Q>Qнаиб)) то
    вывод "ошибка. Введены значения неверны";
    конец программы, return 0;
конец
//строим решето эратосфена:
цел k=0;
for (int i=2; i<=q; i++) {
    int arr[q];      /* Введем локально массив. На некоторое время
                        затраты по памяти будут велики, но зато
                        скорость всей программы больше
                        (не буду затем обрабатывать полученный
                        масив)
                     */
if (arr[i]==0) {
    arr[i]=i; mass[k]=i;
    k++;
    for(int j=i; j<=q; j=j+i) {   //теперь заполним все значения, которые
        arr[j]=i;                 //уже не будут простыми
    }
}
}
/* Наконец мы имеем массив mass[], где хранятся простые числа.
Осталось рассмотреть даные от P до Q числа на принадлежность к B-гладким */
```

[продолжение на след. странице]

```
for (int i = p; i <= q; i++) {
  boole flag = true;
  for (int j = 0; j <= k; j ++) {
    if (p % mass[j] == 0) {   //если p делится без остатка на простое число
                              //из массива mass[],
      if (mass[j] > B) {      //если этот делитель > B
        flag = false;
        break;   //выход из цикла, можно дальше не проверять
                 //т.к. уже условие не выполнилось.
      }
    }
  }
  if (flag == true) {  //если не было ситуации mass[j] > B, то flag = true;
    вывод (i, " - B-гладкие число"); вывод "/n"  //перевод строки, чтобы
                              //отсчитать числа, которые были выведены
}
```

Также, если P = 1, то это число подходит если 1 < B.
Проверим и этот случай:

```
if ((p == 1) && (1 < B)) {
  вывод ("1 - B-гладкое число");
}
```

конец всей программы.

№2.

Простой путь решения данной задачи – перебрать a от 2 до 256 (включительно, не подходит), удостовериться, что $R_{257}(a^i) \neq R_{257}(a^j)$, где $1 \leq j \leq 256$ и не равно i, и проверить, что существует x: $R_{257}(a^x) = 9$; (найти x) проверить, что $\exists y: R_{257}(a^y) = 256$; (найти y)

Мы будем иметь a, x, y и можем вычислить $R_{257}(a^{xy})$;

Но такой способ затратный много памяти и времени, а также при „худшем случае" $a = 256$, тогда 256^{256} очень сильно нагружает вычисления.

[продолжение на след. стр.]

Тогда попробуем другой способ:

давайте попробуем возвести 4 в разные степени.

$4^1 = 4$ $4^4 = 256$. Мы видим, что ~~последующие~~ можно не считать само число

$4^2 = 16$ $4^5 = 1024$ в этой большой степени. Можно каждый раз считать

$4^3 = 64$ $4^6 = 4096$ $R_{257}(4^x)$, его умножать на 4 и считать его остаток от деления.

Давайте тогда писать программу:

```
цел a, x, y, z, t; num
цел mass[258];
если (a > 256) || (a < 0) то
  "Введите значит некорр"
  Конец программы; return 0;
//найдем нужные a!
t=257;
for (int z=2; z<=256; z++){
  num=z;
  for (int i=1; i<=256; i++){      // z - по порядку и под a. т.к. число
                                   // все R257(a^i) разные между собой
    num = (num·z) % t;
    mass[i] = num;
    for(int f=1; f<i; f++){        // Вычисляем, было ли раньше такое
      if (mass[f]==num){           // значение.
        break;  //выходим из цикла, так это z нам не подходит.
      }
    }
    if (num==9){
      x=i;
    }
    if (num=256){
      y=i;
    }
  }
  if((z==256)&&(x!=0)&&(y!=0)){
    a=z;  //мы нашли a, x, y по заданным условиям.
  }
}
```

АЛГОР-М ≠ ПРОГР-МА

Теперь давайте аналогично возведем в степень xy число a:

```
num=a;
for (int i=1; i<=(x*y); i++){
  num=(a*num) % t;
}
```

осталось вывести искомое значение;

вывод "num"

Конец программы

№3.

Давайте рассмотрим, что такое заштрихованная область:
~~Будем~~ Будем считать, что элементы матрицы, лежащие на ~~[illegible]~~ достигаемых нами прямых, параллельных диагоналям, принадлежат заштрихованной области.

(рис 1 – то, что я рассматриваю. рис 2 – не тот случай, что я рассматриваю)

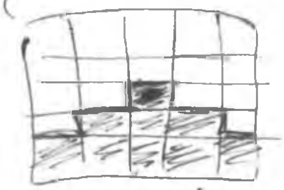


рис 1

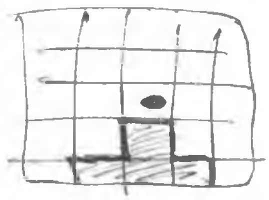


рис 2

• – данный элемент матрицы

```
/* Тогда сначала нужно считать значения в матрицу A:
   аналогично 1 заданию, пусть
   imax=1000; jmax=1000;
   Тогда nmax = 1000 */
цел a[1000][1000];
цел i, j, n, max, x, y, k;

ввод n;
  if ((n<0)||(n>1000)) {
     "введенные данные неверны";
  }
for (int i=1; i<=n; i++) {
  for (int j=1; j<=n; j++) {
       ввод a[i][j];
  }
}
// Теперь вычислим для каждого элемента матрицы max, лежащий
// в заштрихованной области:
for (int i=1; i<=n; i++) {
  for (int j=1; j<=n; j++){
x=j; y=i; k=0; max=0;  //обнуляем max, присваиваем x=j; y=i; (k – длина
                       //отрезка в ряду)
    while (x ≠ (n+1))
        if (a[y][x]>max) {
           max = a[y][x];
      for (z=0; z<=k; z++) {
        if (a[y][x-z] > max)
             max = a[y][x-z];
        }
```

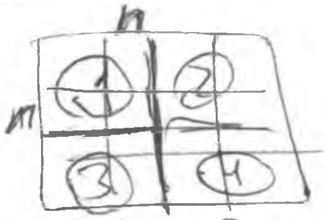
[продолжение на след стр.]

```
if (a[y][x+z] > max)
   max = a[y][x+z];
}
y++;
k++;
}
b[i][j] = max;
}
}
```

Теперь ~~[illegible]~~ выведем матрицу B:

```
for (int i = 1; i <= n; i++) {
  for (int j = 1; j <= n; j++) {
    Вывод(b[i][j], " "); чтобы добавить, чтобы было удобно
  }
  Вывод ("\n"); //перевод строки
}
```

№4



Аналогично введем ограничение на n.
пусть $n_{max} = 1000$;

```
цел num, n; a[2000][2000];
ввод n,
if ((n < 0) || (n > 1000)) {
  вывод "неверные данные"
  или прекратить; return 0;
}
for (int i = 1; i <= n; i++) {
  for (int j = 1; j <= n; j++) {  ввод num;
    if ((i <= n) && (j <= n)) {
      a[i+n][j+n] = num;
    }
    if ((i > n) && (j <= n)) {
      a[i+n][j-n] = num;
    }
```

Алг-м ≠ прог

/* лучшим решением будет сразу во время считывания данных вычислять их в нужное место. ~~то есть~~ Тогда программа не тратит время на 2 отдельных цикла считывания и обработки данных. ~~[illegible]~~ */

/* Я устанавливаю определенные значения в зависимости от попадания их в области; как на рисунке 1↔4; 2↔3; */

[продолжение на след стр.]

```
if ((i > n)&&(j<=n)){
    a[i-n][j+n] = num;
}
if ((i>n)&&(j>n)){
    a[i-n][j-n] = num;
}
}
}
// теперь мы имеем массив, который нужно только вывести:
for (int i=1;i<=n;i++){
    for (int j=1;j<=n;j++){
        printf(a[i][j], " ");  // пробел, чтобы было видно.
    }
    printf("\n");  // перевод строки
}
```

№5

($n \geq 3$)

Нужно определить границы вводимых P, Q, k, проверить на них. (аналогично предыдущим заданиям)

~~Очевидно, что~~ Нам нужно найти все множества $\{n_1, \dots n_k\}$, для которых будет решена задача Знайки.

Можно построить k вложенных циклов, которые будут перебирать все n_i значения для того, чтобы затем каждое число проверить на:

$$\cancel{n_i \ne 0} \quad \prod_{j \ne i}^{k} n_j + 1 \,\%\, n_i \, ; \; n_i \ne \prod_{j \ne i}^{k} n_j + 1$$

~~Асимптотика этой программы велика, но~~

Так рассмотреть все $k \in [P;Q]$.

Асимптотика этой программы велика.

На малых k программа работает за $k^3 \cdot const$.

На больших k время работы будет очень велико.

~~Аналитически можно понять, что в $\{n_1, \dots n_k\}$ не могут входить простые числа, что существенно уменьшит время работы программы~~ ~~Это также уменьшит время~~

Олимпиада школьников «Надежда энергетики»

ИГЭУ

Место проведения

ZS91-46

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ МИРОШНИЧЕНКО

ИМЯ АЛЕКСАНДР

ОТЧЕСТВО СЕРГЕЕВИЧ

Дата рождения 07.05.99

Класс: 11

Предмет ИНФОРМАТИКА

Этап: ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 5 **листах**

Дата выполнения работы: 18.02.17
(число, месяц, год)

Подпись участника олимпиады: [signature]

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-46

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

N1. := - присваивание.

1) Считаем B в переменную B

2) Считаем P и Q в переменные P и Q соответственно.

3) Создадим массив всех простых чисел начиная с 2 до Q. Делаем это следующим образом:

Создаем цикл по i от 2 до Q в нём {

~~проверяем, если i делится хотя бы на~~

проверяем, если i не делится на все числа от 2 до $\sqrt{i}$ то оно простое ⇒ заносим его в изначально пустой массив arr[ ]

}

После цикла у нас есть массив со всеми простыми числами до Q включительно.

4) Создаём временный массив arr1[ ] (пустой) +

5) Создаем цикл по j от P до Q включительно в нём {i:=j

в цикле проверяем i на делимость по каждому элементу из массива arr!

Если остаток при делении i на очередной элемент arr равен 0, то i:= i поделить на arr[~~найденн~~ индекс найденного элемента] и заносим во временный массив arr1[ ]. Делаем всё это пока i не равно 1. (условие 1=i - выход из цикла по занесению чисел в массив arr1

Проверяем теперь в цикле, если каждый элемент массива arr1 меньше или равен B, то выводим, что число "B-гладкое". Иначе выводим, что оно не B-гладкое!

Обнуляем массив arr1, переходя к следующему j. }

6) После работы цикла на экране будут последовательно выведены фразы по порядку для чисел от P до Q, говорящие об их B-гладкости

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-46

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№2

1) Создаём цикл по i от 1 до 256, в нём {
~~Выделяем массив arr[ ];~~ ~~flag = true~~
Создаём массив arr[ ] (пустой).
Создаём цикл по j от 1 до 256, в нём {
занесим остаток при делении i на j в массив arr[ ]
}
Сравниваем теперь в массиве arr[ ] элементы следующим образом (в цикле): {1-ый элемент со всеми остальными от 2-го до 256-го
2-ой элемент со всеми остальными от 3 до 256-го
...
255 элемент с 256 элементом }
В этом цикле ~~если~~ находим одинаковые элементы, то переходим к следующему элементу в цикле по i. Если таковых нет делаем следующее
~~flag = true~~ ~~(конец)~~
Если (i^{256} при делении на 257 даёт остаток 1) ~~&& (flag~~ то {
~~Создаём~~ цикл по k1 от 1 до 256 {
Создаём цикл по k2 от 1 до 256 {
Если (i^{k1} при делении на 157 даёт остаток 9) и (i^{k2} при делении на 257 даёт остаток 256), то { выводим остаток при делении $i^{k1 \cdot k2}$ на 257 как ответ. Завершаем программу }
} } }

}

~~2) Выводим~~

Комментарий: ~~для~~ возведение в степень есть в любом языке программирования.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111 ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-46

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№ 3

1) Считываем значения i в переменную i1, а значение j в переменную j1. Создаём пустой массив arr1 [ ]

2) ~~(Считаем значения n) в переменную n.~~ Создадим массив arr для матрицы.

3) Считаем через двойной цикл от i := 1 до n {
матрицу
от j := 1 до n {
Считать arr[i, j]
}
}

После считывания в двумерном массиве arr у нас есть начальная матрица

~~4) Создаём цикл по i от i1 до n {~~
~~Если i1 = n, то выйти из программы, вывести что заштрихованной области не существ~~

4) Если i1 = n, то вывести, что заштрихованной области не существует и выйти из программы (завершить алгоритм)

5) delta := 0;

6) Создаём цикл по i от i1 + 1 до n {
Цикл по j от (j1 - delta) до (j1 + delta) {
Если (j >= 1) и (j <= n) то arr[i, j] заносим в массив arr1. }
delta := delta + 1;
}

7) После работы цикла в массиве arr1 находятся все элементы заштрихованной области.

8) max := arr1 [0];

9) Создаём цикл по i := 1 до длины массива arr1 {
//Здесь будем искать максимум.
Если arr1 [i] > max { max := arr1 [i] }

10) Вывести значение переменной max в качестве ответа к задаче.

+

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-46

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№ 4

1) читаем значение n в переменную n.
2) Заполняем матрицу в двумерный массив arr[1..2n, 1..2n]
3) Создаём целочисленную переменную w.
4) ~~Создаём цикл~~ {
```
   От i:=1 до n делать {
     От j:=1 до n делать {
       w:=arr[i,j]
       arr[i,j]:=arr[i+n, j+n]        меняем
       arr[i+n, j+n]:=w               элементы
                                      местами
     }
   }
}
```
5) ~~Создаём второй цикл~~ {
```
  От i:=1+n до 2·n {
    От j:=1 до n {
      w:=arr[i,j]
      arr[i,j]:=arr[i-n, j+n]
      arr[i-n, j+n]:=w
    }
  }
```
6) Выводим матрицу конечную.
```
  От i:=1 до 2n {
    от j:=1 до 2n {
      вывести элемент arr[i,j]
    }
    перевод на следующую строку
  }
```

Олимпиада школьников «Надежда энергетики»

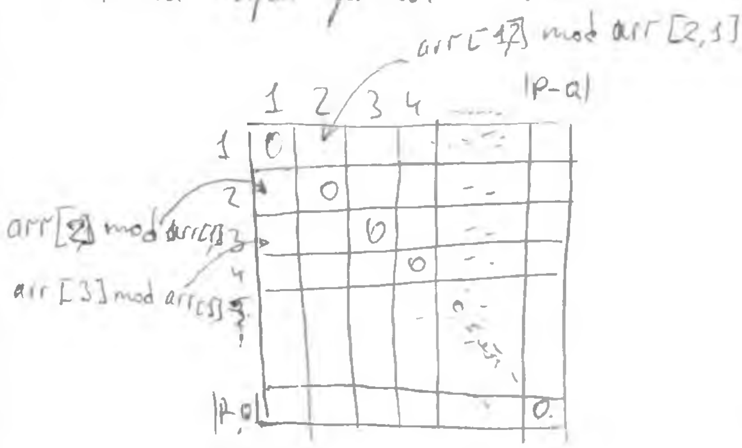
Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-46

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№5.

Будем заносить в матрицу размерности arr[1..|P-Q|, 1..|P-Q|] остатки при делении чисел



mod – остаток при делении

Теорема об остатках:

$$(\underbrace{a\cdot a\cdot a\cdot a}_{n}) \bmod k \Leftrightarrow ((a \bmod k)\cdot n) \bmod k$$

При помощи динамического программирования и формул комбинаторики найдём кол-во способов выбрать k чисел из |P-Q| чисел и сами эти способы.

Затем будем проверять по нашей матрице произведения остатков в ряду матрицы в цикле в каждом способе и сверять, если (произведение остатков в матрице +1) mod k = 0, то тогда к общему количеству решений kol_resh добавим единицу.

В ответ выведем значение переменной kol_resh.

Олимпиада школьников «Надежда энергетики»

ИГЭУ

Место проведения

ZS91-99

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ РОМАНЫЧЕВ

ИМЯ ЛЕОНИД

ОТЧЕСТВО РОМАНОВИЧ

Дата рождения 20.08.1999 **Класс:** 11

Предмет ИНФОРМАТИКА **Этап:** ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 4 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Leon

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111 ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ZS91-99

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

№1.

1. Найдем все простые числа до Q включительно.
2. Теперь для каждого числа(i) в диапазоне от P до Q выполним проверку: а) переберем все простые делители от 1 до i и если все они делят i и при этом меньше или равны B то i-е число - B-гладкое иначе число не подходит, проверяем следующее.

Для нахождения простых чисел воспользуемся алгоритмом «решето Эратосфена».

Алгоритм:

```
int B, P, Q;
cin>>B>>P>>Q;
if (Q<P) cout<<"неверные входные данные";
int A[100000]; l=0, i, j
A[1]=2; l=2;
for (i=4; i<=Q; i+2) used[i]=1;  //вычеркиваем все четные числа
                                  (2 уже запомнили, это простое).
for (i=3; i<=Q; i+=2){
    if(used[i]==0){
        for(j=i+i; j<=Q; j++){
            used[j]=1;
        }
    A[l]=i;  //запоминаем простое число.
    l++;
    }
}
for (i=P; i<=Q; i++){
    j=1; f=0;
    while(A[j]<=i){
        if ( i%A[j]==0){
            if (A[j]>B) {f=1;      //проверка на делимость и
                         break;}   // на то, что простой делитель
        }                          // не превосходит B.
    }
    if (f==0) cout<<"число"<<i<<"B-гладкое";
    ~~else cout~~
}
```

№3

```
int n; cin>>n; if (n<1)
int A[n][n];
for (i=0; i<n; i++){
    for(j=0 j<n; j++){
        cin>>A[i][j];  //ввод матрицы A.
    }
}
```

№3

```
int n, m, i, j, z, p, k;
ввод n;
for i=0 до n {
  for j=0 до n {
    ввод A(i,j); //ввод матриц
  }
}

for i=0 до n {
  for j=0 до n {
    m=0; z=0;
    for k=i до n {
      for p=j-z до j+z {
        if A(k,p)>m { m=A(k,p);
        }
      } z++;
    }
    B(i,j)=m;
  }
}
```

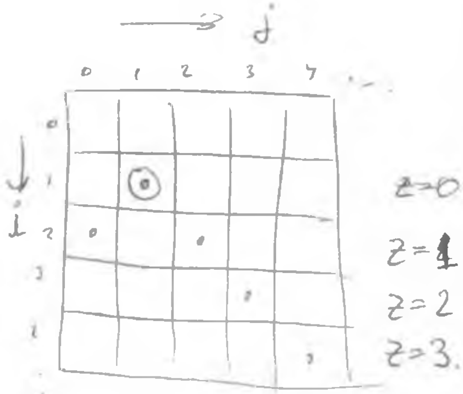


рис. 1.

Будем пол-но заполнять матрицу B.

Например, для $i=1$, $j=1$.

Мы циклами пройдем нижнюю часть и найдем в ней максимум.(переменная m).

При этом количество итераций в первом цикле ровно от i до n

А второго начиная с 1 и увеличивается на 2 каждый раз после 1 итерации 1ого цикла т.к. количество клеток растет с каждой строчкой на 2. (Переменная z).

Нижние части диагоналей проходящие через клетку i и j так же должны учитываться, без них не получится заполнить самую нижнюю строчку матрицы B.

№4.

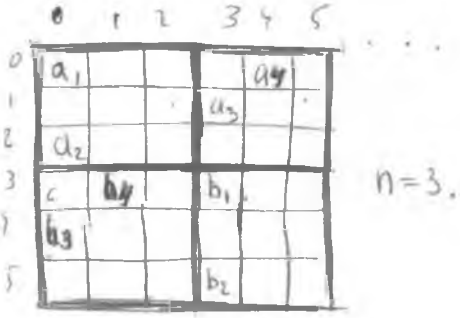
```
for i=0 до 2n {
  for j=0 до 2n {
    ввод A(i,j)     //ввод матриц
  }
}
```

№4 (продолжение)

```
for i=0 до 2n {
  for j=0 до n {
    if (i<n) {
      t=A(i,j);
      A(i,j)=A(i+n,j+n);
      A(i+n,j+n)=t;
    }
    else {
      t=A(i,j);
      A(i,j)=A(i-n,j+n);
      A(i-n,j+n)=t;
    }
  }
}
for i=0 до 2n {
  for j=0 до 2n {
    вывод A(i,j);
  }
  перевод строки;
}
```

Алг-м + прогр

+



n=3.

рис 1.

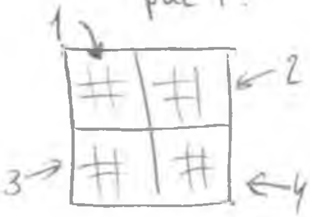


рис. 2.

Пояснение: обозначим квадраты 1, 2, 3, 4 как показано на рис 2.

Теперь перебирая ячейки исходной таблицы действуем в соответствии с тем, в каком квадрате находится ячейка:

- Если в 1м, то меняем местами ячейки A(i,j) и A(i+n,j+n). Например: a_1 и b_1; a_2 и b_2. (рисунок 2)
- Если во 2м, то меняем A(i,j) и A(i-n,j+n). Например: a_3 и b_3; a_4 и b_4 (рисунок 2).
- Если в квадрате 3 или 4 - ничего не делаем, обмен уже произошел.

№2

Число a равно 256. Т.к. при возведении его в i от $1 \le i \le 256$ остатки от деления уменьшаются на 1. (256, 255, ... 1).

Теперь найдем числа x и y, зная $R_{257}(a^x)=9$, $R_{257}(a^y)=256$.

Очевидно y=1.

Переберем степени x от 1 до 256 и найдем x:

```
f=1;
for i=1 до 256 {
    f*=a;
    if (a%257==9) {x=i;
                   break;}
}
```

+

№ 2 продолжение:

Скорее всего при нахождении числа x у нас произойдет переполнение типа. Но мы можем найти $R_{257}(a^{xy})$ и без числа x, т.к. $y=1$, то $R_{257}(a^{xy}) = R_{257}(a^{x}) = 9$ (по условию). Сл-но секретное число 9.

№5.

Для $k=1$ решений нет.

Для $k=2$ решений 10^{15} т.к.

$$\begin{cases} 1 & 1 \\ 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ \vdots \\ 10^{15}-1; & 10^{15} \end{cases}$$

Для $k=3$ получим решения из $k=2$. Например, если $k=2$ $(2,3)$ то для $k=3$ $(2,3,2\cdot3+1)$.

$k=2$ $(3,4)$ то для $k=3$ $(3,4,3\cdot4+1)$

Т.е. зная решение для k можно найти решение для $k+1$, переберем все множества для k, в каждом из них найдем произведение всех членов и прибавим 1. Тогда если это число меньше 10^{15}, то данное мн-о $k+1$ можно увеличить на 1 добавив все элементы рассматриваемого мн-а (их k штук) и добавим найденный элемент.

```
cin>>P>>Q;
```

Найдем число решений для P.

//Для $k=2$ число решений $\sqrt{10^{15}}$, т.к. решения вида

```
for i = 2 до P {
    p = sqrt(p);
}

for i = P до Q {
    вывод p;
    p = sqrt(p);
}
```

$(1,2)$ $(2,3)$ $(3,4)$ будут продолжаться пока $a\cdot b < \sqrt{10^{15}}$.
Но и для $k=3$ решения (a,b,c) имеют вид $(a\cdot b, c)$ где $c - a\cdot b = 1$, тогда их опять же квадрат числа решений для $k=2$.

Т.е. чтобы найти решения для Z нужно Z раз вычислить $\sqrt{k}$, где k - верхняя граница чисел в множестве.

+

Олимпиада школьников «Надежда энергетики»

МБОУ „СОШ №2"
г. Гусь-Хрустальный

Место проведения

ГК46-48

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ Смирнов

ИМЯ Кирилл

ОТЧЕСТВО Сергеевич

Дата рождения 10. 06. 1999

Класс: 11

Предмет информатика

Этап: заключительный

Работа выполнена на 4 **листах**

Дата выполнения работы: 18. 02. 2017
(число, месяц, год)

Подпись участника олимпиады:

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ГК46-48

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

(1) 1) запускаем цикл по i (от P до Q) – чтобы проверить каждое число на в-гладкость

2) если число i натуральное, тогда проверим его на данное свойство:

если число i четное, тогда в переменную j положим i-1, иначе положим i в j.

пока j > 0 делаем:
если i делится на j без остатка, тогда проверим его (число j) на простоту, для этого заведем специальную функцию и отправим туда j;

(функция) проверка на простоту

в переменную k положим j; в перем. S положим 0 (S=0)

l – кол-во делит. числа j

l = 2, далее запустим цикл по l.
пока (l <= √j и S равно 0) делаем:
если число k делится на l без остатка, тогда
S = 1;
l увеличим на 1.

функция возвращает знач. S. (если S = 0 => число j простое, иначе j не явл. простым)

продолжение:
если S = 0 => j – наибольший простой делитель числа i

=> если j <= в, тогда 1. число i явл. в-гладк. 2. j = -1 (чтобы выйти из цикла по j)

если j > в, тогда 1. число i не явл. в-гладким 2. j = -1

Если мы не нашли простого наиб. делителя (если i не делится на j без остатка или j не простое) => j увелич. на 2 (два) и

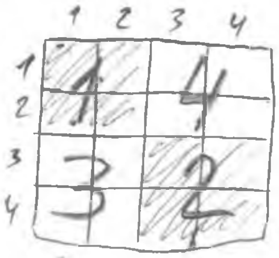
N5-нет

(7)

ищем след. j (проверяем след j на простоту и на делимость i на j)
пока не найдем наиб. простого делителя;

если мы не нашли после всех выполненных операций наиб. простого делителя => проверим простое четное число 2 по пред. алгоритму т.к. оно тоже явл. простым и a[i] может делиться на 2.

(4)

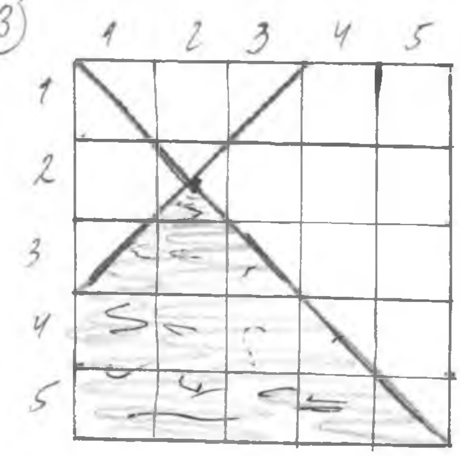


дана матрица a, размером 2n на 2n
пронумеруем строки и столбцы, начиная с 1,
a[i][j] – обращение к элементу матрицы с номером строки i, номером столбца j.

переставим местами блоки 1 и 2:
для i = 1 ... n:
для j = 1 ... n:
меняем местами элементы a[i][j] и a[i+n][j+n]

переставим местами блоки 3 и 4:
для i = (n+1) ... 2n:
для j = 1 ... n:
меняем местами элементы a[i][j] и a[i-n][j+n].

3



Дана матрица A $(n*n)$;
пронумеруем строки и столбцы матрицы, начиная с 1;

A[i][j] - обращ. к элементу матрицы с номером строки i и номером столбца j.

Заведем матрицу B такого же размера, что и A и будем постепенно ее заполнять.

```
для i = 1 .. n :
для j = 1 .. n :   (перебираем все элементы матр. A, чтобы заполнить матр. B)
нач.
    max = -∞ (макс. элемент из нужной нам части)
    s = 0; (переменная необходимая нам для поиска max в заштрих. области)
    для k = i .. n :
        если j - s <= 0, тогда l = 1, иначе l = j-s;
        если j+s > n, тогда r = n, иначе r = j+s;
        для t = l .. r :
            если A[k][t] > max, тогда max = A[k][t];
        s = s + 1;
    B[i][j] = max;
кон.
```

Матрица B заполнена!

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

② 1) подберем числа a, которые удовлетворяют условию задачи; заведем динамический массив, в кот. будет добавлять все числа, кот. могут быть секретными.

пустой динам. массив r

заведем массив ost на (256+1) элементов, изначально заполнен нулями. (нумерация массивов с нуля)

для a = 1 ... 256 делаем:

- s = 1; v = 0; x = 0; y = 0
- для i = 1 ... 256 делаем:
  - s = s · a;
  - s = s % 257;
  - если s = 9, тогда x = i;
  - ost[s] = ost[s] + 1; (кол-во чисел с данным остатком от деления (s))
  - если s = 256, тогда y = i;
  - если (i = 256 и s = 1), тогда v = 1.
- d = 0;
- для i = 0 ... 256 делаем:
  - если ost[i] ≠ 1, тогда d = 1.
- если (V = 1 и d = 0)
  - если (x ≠ 0 и y ≠ 0)
    - → $\boxed{}$
- очищаем массив ost, полагая все значения 0.

% означает остаток от деления

x % y = ост. от деления x на y

s1 = 1

для k = 1 ... x·y делаем:

- s1 = s1 · a;
- s1 = s1 % 257;

добавляем в массив r число s1.

ЗАЧЕМ?

В результате в массиве r лежат все числа, которые могут быть секретными.

Олимпиада школьников «Надежда энергетики»

МБОУ „СОШ №2"	ГК46-41
Место проведения	шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37111

ФАМИЛИЯ ФЕДЯШКИН

ИМЯ МАКСИМ

ОТЧЕСТВО АЛЕКСЕЕВИЧ

Дата рождения 18.11.1999 **Класс:** 11

Предмет ИНФОРМАТИКА **Этап:** ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 4 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады: Федяшкин

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37111

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ГК46-41

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

① Программа №1

Объявление переменных B, P, Q - целые числа

j, m, n, i, t, k - целые числа

Начало программы

ввод (B, P, Q);

для i = P до Q делать начало (1)

m := i; n := B; k := 0

пока n^2 <= m делать начало (2)

n := n+1; t := 0;

для j = 1 до n делать

если n делится на j без остатка, то t := t+1;

Если t = 2 и m уточнено делится (без остатка) на n, то

k := k+1;

конец (2);

Если k = 0, то выводим (i, 'является B-гладким числом');

конец (1);

конец программы

② Программа №2

Объявление переменных p, p', p², p³, x, y, a, l : ~~целые~~ натуральные числа

j, i, m, n, s, k : натуральные числа

Начало программы

{найдем A}

s := 0; a := 1;

Пока s <= 0 делать начало (1)

a := a+1; m := 1;

для i = 1 до 256 делать

m := m • a;

для p = 1 до 256^{256} делать

если 257p + 1 = m, то s := s+1

конец (1);

{найдем x}

s := 0; x := 0;

Пока s <= 0 делать начало (2)

см. на след. листе

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

x := x+1 ; m := 1;
для i := 1 до x делаем
m := ~~m~~ m · a;
для p1 := 1 до 256^{256} делаем начало(3)
если m = 257 · p1 + 9 , то s := s + 1;
конец(3);
конец(2);
{найден y}
s := 0; y := 0;
Пока s <= 0 делаем начало(4)
y := y+1; m := 1;
для i = 1 до y делаем
m := m · a;
для p2 = 1 до 256^{256} делаем начало(5)
если m = 257 · p2 + 256 , то s := 1;
конец(5);
конец(4);
{найден секретное число}
k := x · y; m := 1;
для i = 1 до k делаем
m := m · a;
p3 := m div 257;
l := m - p3 · 257;
вывести (l, является секретным числом)
конец программы.

③ Программа №3
Объявление переменных A - массив [1..n; 1..n] с числами целого типа;
n const = n;
B - массив [1..n; 1..n] с числами целого типа;
x, i, j, m, k, p, max : числа целого типа

Начало программы
для i := 1 до n делаем начало(1)
для j := 1 до n делаем начало(2)
см. на следующем листе

~~max:=A[i, j]~~ max:=A[i;j]

m:=i; k:=j p:=j;

Пока m<n делаем начало(3)

m:=m+1; k:=j-1; p:=p+1;

если p>n, то p:=n;

если k<1, то k:=1;

для x от k до p делаем

если a[m,x]>max тогда max:=a[m,x];

если n=m, то b[i;j]:=max;

конец(3)

конец(2)

конец(1);

конец программы.

(4) Программа N4

Объявление переменных: Nconst=n

A - массив [1..2n, 1..2n] с целыми числами

i, j, m, x, y: целые числа

Начало программы

для i=1 до n делаем начало(1)

для j=1 до n делаем начало(2)

m = A[i;j];

A[i;j]:=A[i+n; j+n];

A[i+n; j+n]:=m;

конец(1)

конец(2)

для i=n+1 до 2n делаем начало(3)

для j=1 до n делаем начало(4)

m:=A[i;j];

A[i;j]:=A[i-n; j+n];

A[i-n; j+n]:=m;

конец(3)

конец(4)

конец программы.

⑥ Пусть n_1, n_2, n_3, n_4, n_5 - последовательность 3ших, тогда должно соблюдаться равенство

$$\frac{n_1 \cdot n_3}{n_i} + \frac{1}{n_i}$$

Возможны случаи

1) $n_i = \pm 1$, тогда вся последовательность будет состоять из единиц или минус единиц (проверка $\frac{-1 \cdot -1}{-1} + \frac{-1}{-1} = 2$ верно)

2) если n_i - четное число, тогда равенство соблюдается если $n_1 \cdot n_3$ - является нечетным числом как и 1, но т.к. n_i может быть и n_1 и n_3, то n_1 и n_3 - всегда четные и их произведение также четное число, следовательно n_i - не может быть четным. Это верно и для отриц. чисел (n_i может быть отриц.)

3) Если n_i - нечетное число, и $n_i \neq 1$ тогда равенство соблюдается если $n_1 \cdot n_3$ также является четным числом, но т.к. n_i может быть и n_1 и n_3, то n_1 и n_3 - всегда нечетные и их произведение также всегда нечетным, что противоречит условию $n_1 n_3$ - четное произведение, следовательно n_i не может быть нечетным кроме единицы.

4) Если $n_i = 0$, то невозможно поделить единицу на 0, следовательно $n_i \neq 0$

Итак n_i - может принимать только 2 значения +1 и -1

~~Программа №5~~

~~Начало г~~ ~~Обозначение переменных~~

Программа №5

обозначение переменных K, P, Q : целые числа.

m: целое число.

Начало программы

m:= Q - P;

Вывод (m * 2 ! кол-во возможных вариантов).

конец программы.

Олимпиада школьников «Надежда энергетики»

Место проведения

ФЫ 31-23

шифр

← **Не заполнять**
Заполняется ответственным работником

Вариант № 37101

ФАМИЛИЯ ФИЛИППЕНКО

ИМЯ ВЕРОНИКА

ОТЧЕСТВО ВИКТОРОВНА

Дата рождения 24.09.2000 **Класс:** 10

Предмет ИНФОРМАТИКА **Этап:** ЗАКЛЮЧИТЕЛЬНЫЙ

Работа выполнена на 3 **листах** **Дата выполнения работы:** 18.02.2017
(число, месяц, год)

Подпись участника олимпиады:

Впишите свою фамилию имя и отчество печатными буквами, дату рождения, класс, название предмета, этапа Олимпиады, общее количество листов, на которых выполнена работа и дату выполнения работы.

Олимпиада школьников «Надежда энергетики»

Вариант: 37101

ШИФР, НЕ ЗАПОЛНЯТЬ! ⇨ ФБ31-23

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

#1
```
алг номер 1;
арг (B, P, Q : ~~цел~~; i : цел);
рез B, i;
нач
  вывод('введите B, P, Q');
  ввод(B, P, Q);
  для i от P до Q
  нц
    если (i <= B) и ((B mod i = 0))
    то
    вывод('число', i, ' - ', B, '-гладкое');
  кц;
кон;
коналг.
```

+

#4
```
алг номер 4;
арг ( ~~x[2N,2N]~~ x[2N,2N] : целтаб; n, i, j : цел; buf : цел);
рез (~~...~~ ; (x)2N,2N ;
нач
  вывод('введите n');
  ввод(n);
  для i от 1 до 2n
  нц
    ~~вывод('введите~~
    для j от 1 до 2n
    нц
      вывод ('введите x[', i, ',', j, ']');
      ввод (x[i, j]);
    кц;
  кц;
  для i от 1 до n
  нц
    для j от 1 до n
    нц
      buf1 := x[i, j+n];
      x[i, j+n] := x[i, j];
      buf2 := x[i+n, j+n];
      x[i+n, j+n] := buf1;
      buf1 := x[i+n, j];
      x[i+n, j] := buf2;
```

можно: ввод (X[2N,2N]); (вроде)

?

±

см. далее →

ВНИМАНИЕ! Проверяется только то, что записано с этой стороны листа в рамке справа

```
x[i,j]:=buf1i
кц;
кц;
для i от 1 до 2n
нц
для j от 1 до 2n
нц
вывод(x[i,j]);            } можно: „вывод(X[2N,2N]);" (вроде)
кц;
кц;
кон;
конalг.

#3 алг номер3;
арг (a[n,n];b[n,n]:четвоб; min,k,l,i,j,n:цел;);
рез (b)n,n;
нач
вывод('введите n');
ввод(n);
для i от 1 до n
нц
для j от 1 до n
нц
вывод('введите a[',i,',',j,']');      } -//-, см #4
ввод(a[i,j]);
кц;
кц;
для i от 1 до n
нц
для j от 1 до n
нц
min:=a[i,i];
для k от i до j                       ±
нц
для l от i до k
нц
если min>a[k,l]
то min:=a[k,l];
кц;
кц;
b[i,j]:=min
кц;
кц;
```

см. далее →

ВНИМАНИЕ! Проверяется только,то, что записано с этой стороны листа в рамке справа

```
для i от 1 до n
нц
для j от 1 до n
нц
вывод(b[i,j])
кц
кц;
кон;
коналг.
```

}-//-, см #4}

#5.
```
алг номер5;
арг (k, P, Q, m:цел; i, j:цел;);
рез m;
нач
вывод('введите k, P, Q');
ввод(k, P, Q);
m:=0;
для i от P до Q
нц
для j от P до Q
нц
если (i<>j) и ((∏_{i≠j}^{k} * j + 1) mod i = 0) и ((∏_{=j}^{k} * j+1) ≠ i)
то m:=m+1;
кц;
кц;
вывод(m);
кон;
коналг.
```

#2. В данной задаче некорректно поставлено условие и не верны известные данные, так как $R_{257}(a^y)$ не делится нацело на $R_{257}(a^x)$ и наоборот => a не натуральное ч-ло, так же это доказывает тот факт, что a^{256-x} и $a^{256-y} < 1$.