Олимпиада школьников «Гранит науки»
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ

Шифр работы Ш38-266
(не заполнять)

ВАРИАНТ № 3

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
Первая проверка	Полученный балл	0	–	10	0	24	20					54
	Σ баллов прописью	пятьдесят четыре (54)						Подпись проверяющего				
								Подпись проверяющего				
Вторая проверка	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
	Полученный балл	0	–	10	0	24	20					54
	Σ баллов прописью	пятьдесят четыре (54)						Подпись проверяющего				
								Подпись проверяющего				

Олимпиада школьников «Гранит науки» Шифр работы И38-266
(не заполнять)
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ

Вариант № 3

105

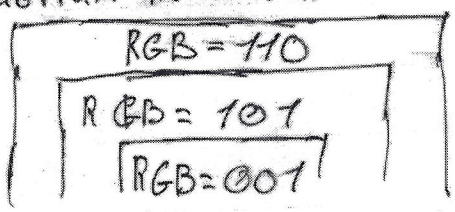
3.

X	Z	Y	$\bar{X} \wedge Y$	$\bar{Z} \vee X$	$\bar{Y} \vee Z$	$F(X,Y) \to F(X,Z)$	$F(Y,Z) \wedge G(X,Y,Z)$
0	0	0	0	1	1	1	1
0	0	1	1	1	0	1	0
0	1	0	0	0	1	1	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	1	1
1	0	1	0	1	0	1	0
1	0	0	0	1	1	1	1
1	1	1	0	1	1	1	1

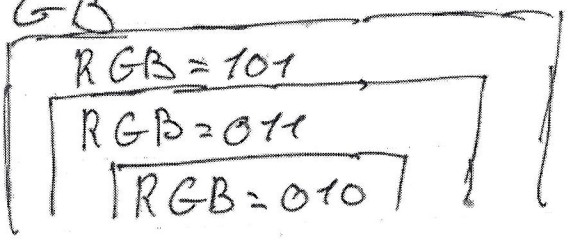
4. Пусть, максимум - 1, а минимум - 0

Рассмотрим ~~первый~~ картинки в RGB

I



II



То есть 110 → 101
101 → 011
001 → 010

Очевидно заметить, что алгоритм представляет собой сдвиг всех значений влево

10

5. яп C++

~~#include <iostream>~~
~~using namespace std;~~
~~int main()~~
~~{~~

```
#include <iostream>
using namespace std;
const long long Max1 = 10e18; // максимально возможное число
bool plain[Max1]; // массив отражающий решето эратосфена
long long ~~int~~ round[Max1]; // массив позиций людей относительно изначального круга
long long l_plain = 1 // будет хранить последнее простое число ГУП
int main()
{
```

Вариант № 3

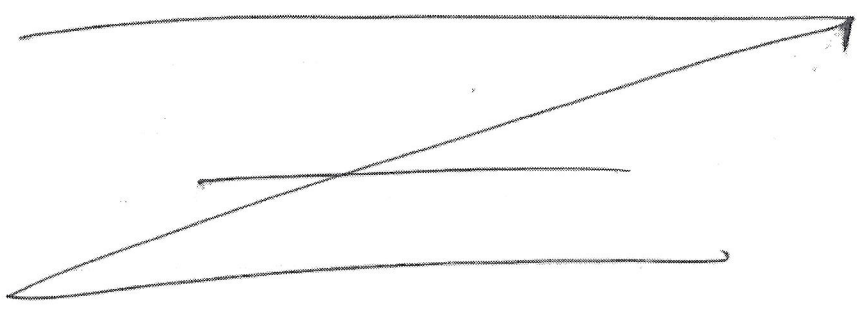
```
for ( long long i=0; i<MaxI; ++i)
{plain[i]= true; round[i]=i;} // заполнение массивов (подготовка)
for (long long i=0; i<MaxI; ++i)
{for(long long j=i·2; j<MaxI; j+=i)
 {plain[i]=false;}} // реализация решета Эратосфена,
                    ложные элементы — не простые
long long N,M, G;
cin>> N>>M>> G;
for (long long r=0; r<M; ++r) // каждый раунд
{ int p=0; // изначально ставлем от нашей позиции 0
  do
  {l_plain++;} while (!plain[l_plain]); // ищем следующее простое число
  for(long long i=0; i<l_plain; ++i)
  { if (p== N-1) // обработка случая конца круга
    { long long t = round[p];
      round[p]=round[0];
      round[0]= t;
      p=0
    } else // обработка остального случая
    { long long t=round[p];
      round[p]=round[p+1];
      round[p+1]= t;
      p+=1;
    }
  }
}
for (long long i=0; i<MaxI;++i) // ищем G
```

Вариант № 3

```
{ if ( round[i] == G-1) и находим человека
  { if (i==0) // если он в начале круга
    { cout<< round[1]<<,0'<< round[N-1]; }
    else if (i == N-1) // если в конце
    { cout<< round[0]<<,0'<< round[N-2];
    else
    { cout<< round[i+1]<<,0'<< round[i-1];
    return 0; // сделали всё, что надо завершаем работу
  }
}
```

246

Простые числа для ходов будем находить с помощью решета Эратосфена. Будем хранить, какой человек стоит на ~~тая~~ позиции круга, т.е. round[2] Таким образом в начале каждого раунда будем менять местами с round[1] справым от него человека на позиции 0 round[2] l_plain раз, где l_plain — простое число по порядку раунда. Обрабатывая проблему перехода между последним и первым элементом ~~все должно обрасотать~~ проходим все раунды. В итоге остаётся найти в итоговом массиве позиций искомого человека и вывести его соседей.



Олимпиада школьников «Гранит науки» **Шифр работы** И38-266 (не заполнять)

БЛАНК ОЛИМПИАДНОЙ РАБОТЫ

Вариант № 3

~~1. | A | B | C | D | E~~

~~A dBe | b 2 b | d 7 a~~

A3 − 1 = 4 A3 = 5

5 + B3 + 4 = 11 B3 = 2

2 − 2 + C3 = 1 C3 = 1

1 − D3 − E3 = −6 → D3 = 7 − E3

1 + D3·E3 = 7 → 1 + (7 − E3) E3 = 7 → $E3^2 - 7E3 + 6 = 0$

E3 = 6, D3 = 1

E3 = 1, D3 = 6

a b c d e f g
1 2 3 4 5 6 7

A4 = „bcadfe" F4 = d
1 2 3 4 5 0

A4 = C4 = d

abcd => B4 = b

B4 C4

E4 + D4 = 9 → E4 = 9 − D4

E4 · D4 = 20 → (9 − D4)·D4 = 20 →

→ $D4^2 - 9D4 + 20 = 0$

D4 = 5, E4 = 4

D4 = 4, E4 = 5

a b c d e f
0 1 2 3 4 5

A1 = de B1 = bb C1 = da

D1, E1 → fa, fe
ff, ae

↓

ea, fa
ef, af

00

Вариант № 3

6. Решение задачи можно осуществить через рекурсию.
Устройство рекурсивной функции.
1. Считаем число свечей, которое уже передано из прошлой рекурсии.
. если ~~четное~~ равно 1, увеличиваем число ответов, выходим
. если нечётное, выходим
2. если размеры торта по горизонтали четны:
считаем число свечей в каждой из частей:
. если равно, запускаем рекурсии от каждого из кусков
3. если размеры торта по вертикали чётны:
если число свечей в каждой из частей равно,
запускаем ~~от~~ рекурсию от каждого из кусков
Выходим
Выводим ответ

~~Код на~~ ~~ЯП~~ Python

~~def rec (x1, y1, x2, y2, n)~~

~~if n%2==0~~

~~if n==~~

Код на ЯП C++

```
#include <iostream>
using namespace std;
long long Max1 = 10e18;
bool long long cake[Max1][Max1]; // массив торта, где true - свеча
long long ans = 0;
void rec ( long long x1, long long y1, long long x2, long long y2, long long n) // координаты границ куска и число свечей, где правая и нижняя границы невключительны.
{ if (n==1)
  { ans++; return; }
  if (n%2==1)
```

15+5=20

Вариант № 3

```
{return;}
if ((x2-x1)%2==0) // делим вертикально
{ long long n1=0, n2=0; // число свечей в половинке
  for(long long i=x1; i<(x2+x1)/2; ++i)
    {for(long long j=y1; j<y2; ++j)
     { n1+=cake[i][j]; }} // суммируем число свечей
  for(long long i=(x2+x1)/2; i<x2; ++i)   на левом куске
    {for{ long long j=y1; j<y2; ++j)
     { n2+=cake[i][j]; }} // свечи правого куска
  if (n1==n2)
  { rec(x1,y1,(x2+x1)/2,y2, n1);
    rec((x2+x1)/2, y1, x2, y2, n1); }}
if ((y2+y1)%2==0) // делим горизонтально
{ long long n1=0, n2=0;
  for(long long i=y1; j=(y1+y2)/2; ++i)
     ... аналогично, как с верт. разрезом
  ...
}
return;
}
int main()
{ long long n, N, k;
  cin>
```