Олимпиада школьников «Гранит науки» Шифр работы 438-188
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

ВАРИАНТ № 5

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
Первая проверка	Полученный балл	—	3	7	20	15	7					52
	Σ баллов прописью	пятьдесят два (52)						Подпись проверяющего		Ивн		
								Подпись проверяющего				
Вторая проверка	№ задания	1	2	3	4	5	6	7	8	9	10	Σ
	Полученный балл	—	3	7	20	15	15					60
	Σ баллов прописью	шестьдесят (60)						Подпись проверяющего				
								Подпись проверяющего				

+8

Решение на след. листе

Олимпиада школьников «Гранит науки» Шифр работы У38-188
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 5

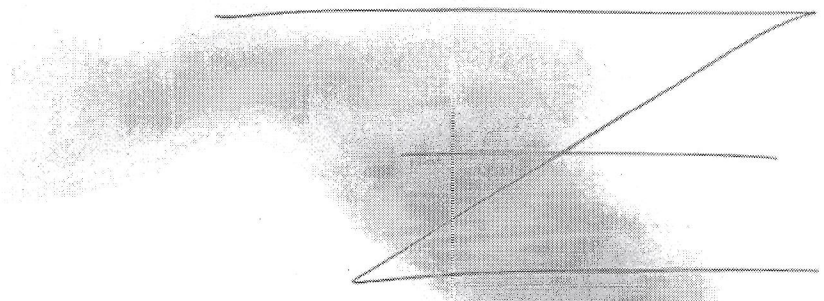
№5

```
#include <iostream>
using namespace std;
int main(){
    int n, r, d;
    cin >> n >> r >> d;
    long long MOD = 1000000007;
    long long R = 1;
    long long G = 0;
    while (d > 0){
        G = G + R(n - r);
        G %= MOD;
        R = R * r;
        R %= MOD;
        d -= 1;
    }
    cout << R << " " << G << endl;
    return 0;
}
```

155

15 баллов

150

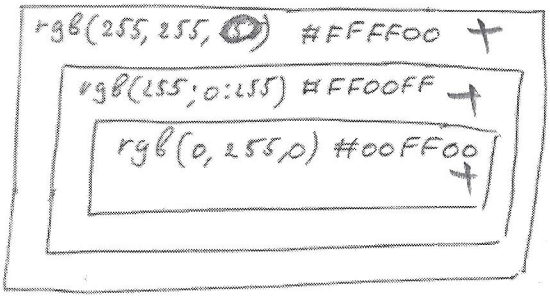


Вариант № 5

№ 4

Алгоритм работает таким образом, что в каждом канале "инвертирует" значение. То есть если в исходном изображении был пиксель с цветом (x; y; z) в цветовом пространстве RGB (x – яркость красного канала, y – яркость зеленого канала, z – яркость синего канала), то обработанном изображении этот пиксель стал бы (255 – x; 255 – y; 255 – z). Алгоритм просто применяет эту операцию для каждого пикселя в изображении.

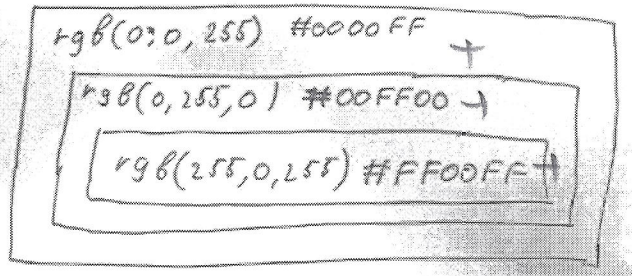
Рисунок 1



2 б 2 б

2 б

Рисунок 2



Олимпиада школьников «Гранит науки» Шифр работы И38-188
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

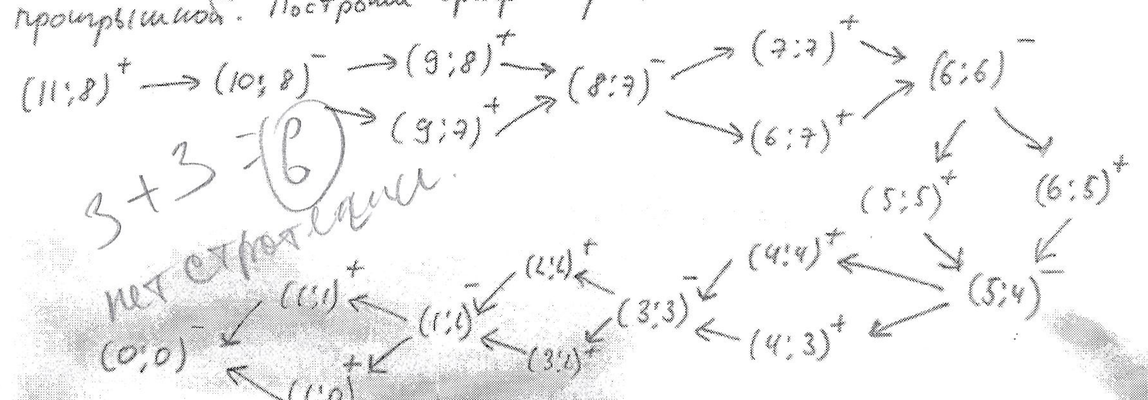
Вариант № 5

№2

~~Главному выгодно ходить так, чтобы после его хода камней в кучах было четное кол-во (в сумме). Таким образом на каждый ход~~

~~Главному выгодно ходить так, чтобы~~ ~~камней в кучах было~~

Стратегия игры с 11 и 8 камнями: будем за $(n;m)^+$ — обозначать выигрышную позицию с n и m камнями в кучах и за $(n;m)^-$ — проигрышную. При этом позиции $(n;m)$ и $(m;n)$ считаем идентичными. Оказавшись в проигрышной позиции игрок проигрывает, то есть как бы он не ходил, противник будет в выигрышной позиции. Выигрышная позиция если из нее можно перейти в проигрышную. $(0;0)$ считаем проигрышной. Построим граф игры:



Таким образом выигрывает первый т.к. изначально находится в выигрышной позиции. Стратегия на графе.

ЛИСТ 4 из 8

3б

3б

Вариант № 5

№3

Т.к. в конце схемы стоит или, то левая и правая часть должна быть 0

$$\begin{cases} x_2 = 0 \\ (x_1 \text{ и } x_2) \text{ xor } x_3 = 0 \end{cases} \Leftrightarrow \begin{cases} x_2 = 0 \\ x_1 \text{ - любое} \\ 0 \text{ xor } x_3 = 0 \end{cases} \Leftrightarrow \begin{cases} x_2 = 0 \\ x_1 \text{ - любое} \\ x_3 = 0 \end{cases}$$

Т.к. $x_2 = 0$, то x_1 может быть любым, ведь x_2 и $0 = 0$ $\forall x_1$

0 xor $x = 0$ только если $x = 0$

Итого имеем наборы:

x_1	x_2	x_3
0	0	0
1	0	0

1+2 наборы:

x_1	x_2	x_3
0	0	1
1	0	1

Выражение:

$\{x_1 \to \overline{x_2}\}$ и $\{x_3$ или $x_1\}$ — считало, что символ ⊕ обозначает логическое и, потому что в задаче расшифровка не написана

x_1	x_2	x_3	R
0	0	1	1
1	0	1	1

Ответ: ↗

2б

7б описки

7б

лист 5 из 8

Олимпиада школьников «Гранит науки» Шифр работы Ч38-188
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 5

#6

```
#include <iostream>
#include <vector>
#include <string>
using namespace std;

int W, H;
vector<vector<bool>> t; vector<vector<int>> memo;
vector<vector<bool>> visited;

bool next_move(int posx, int posy, int size) {
    for (int x = posx; x < posx+size && posx+size < W) {
        for(int y = posy; y < posy+size && posy+size < H) {
            if (!t[x][y]){
                return false;
            }
        }
    }

    if (x+posx >= W){
        return true;
    }
    if (memo[posx][posy]!=0) return memo[posx][posy]-1;
    bool can = false

    if (posx+1 < W && !visited[posx+1][posy]) {
        visited[posx+1][posy] = true;
        can = next_move(posx+1, posy, size);
        visited[posx+1][posy] = false;
    }
```

78

78

~~70~~

155

Олимпиада школьников «Гранит науки» Шифр работы У38-188
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 5

№6 (продолжение)

```
    if (!can && posy+1 < H && !visited[posx][posy+1]) {
        visited[posx][posy+1] = true;
        can = next_move(posx, posy+1, size);
        visited[posx][posy+1] = false;
    }
    if(!can && posx-1 >=0 && !visited[posx-1][posy]){
        visited[posx-1][posy] = true)
        can = next_move(posx-1, posy, size);
        visited[posx-1][posy] = false;
    }
    if (!can && posy-1 >=0 && !visited[posx][posy-1]){
        visited[posx][posy-1] = true;
        can = next_move(posx, posy-1, size);
        visited[posx][posy-1] = false;
    }
    memo[posx][posy] = can+1;
    return can
}

int main() {
    cin >> H >> W;
    t.resize(W, vector<bool>(H));
    visited.resize(W, vector<bool>(H));
    for(int y=0; y<H; y++){
        string s; cin >> s;
        for(int x=0; x<W; x++){
            t[x][y] = (s[x] == '.');
        }
```

memo.resize(W, vector<int>(H));

Вариант № 57

№6 (продолжение)

```
ans int ans = 0;
for (int r = 1; r < H-1; r++){
  for( int y = 1; y < H-1; y++){
    if (next_move(0,y,r)){
      ans = r;
      break;
    }
  }
}
for(int x=0; x< W; x++){
  for( int y=0; y< H; y++){
    memo[x][y]=0
  }
}
cout << ans << endl;
return 0;
}
```