Олимпиада школьников «Гранит науки»

БЛАНК ОЛИМПИАДНОЙ РАБОТЫ

Шифр работы 408-15

(не заполнять)

Вариант № 3

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

№ задания		1	2	3	4	5	6	7	8	9	10	Σ
Полученный балл		—	10	—	12	25	2					49
I проверка	Фамилия И.О. проверяющего	Кочева А.А			Подпись			Σ баллов прописью	сорок девять			
	Фамилия И.О. проверяющего	Кива М.Р.			Подпись			Σ баллов прописью	сорок девять			
II проверка	Фамилия И.О. проверяющего	[illegible]			Подпись			Σ баллов прописью	Сорок девять			
	Фамилия И.О. проверяющего				Подпись			Σ баллов прописью				

Вариант № 3

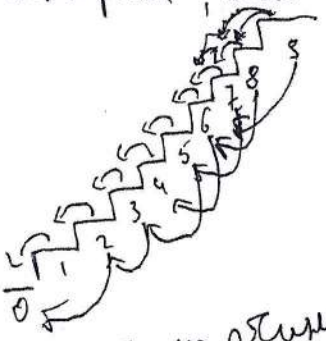
№2

~~Выиграет 1-ый игрок:~~

Пусть он возьмёт 3 конфеты из 27 =) ~~останется 24~~

№5

Давайте рассмотрим каждую ступеньку отдельно и посмотрим, как в неё ~~*~~ можно попасть.



По данной картинке можно понять, что в этой задаче применяется метод динамического программирования, где $dp[1]=1$ $dp[2]=2$ $dp[i] = dp[i-1] + dp[i-2]$ $i>2$

1 2 3 4 5 6 7 8 9
1 2 3 5 8 13 21 34 55

Ответ: 55

Как нам получить ответ:

Давайте хранить 2 предыдущих значения и получать нужное. Также есть другой способ. Если n очень большое, мы можем решить эту задачу путём ~~возведе~~ быстрого возведения матрицы $\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ в степень, нужную нам. У этого решения асимптотика будет $\log_2 n$.

```
#include <bits/stdc++.h>
using namespace std;
signed main() {
    int a=1, b=2, n; // наши переменные
    cin >> n;
```

Вариант № 3

```
n=n-2;
while(n--) {
 int c=a+b; // наше значение на i шаге
  a=b%a;
  a=c;
}
cout<<a;
}
```

250

№ 2

~~В этой игре всегда победит 1 игрок. Общий алгоритм:~~

Давайте опишем общий алгоритм.

Пусть цифра 1 - победит 1-ый, -1 - 2ой.

Очевидно, что если из позиции i, есть ход, где в -1, то в эту позицию можно записать 1 (т.к если сходить в позицию с -1, то 2-ой точно проиграет), а если нет, то -1.

i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	-1	1	1	1	1	-1

Из позиций 1, 2, 3, 4 1-ый игрок точно победит, а для остальных приведён выше описанный алгоритм. Заметим, что если

i : 5, то победит 2, иначе 1.

27 % 5 => победит 1-ый

100

№ 4

Вариант № 3

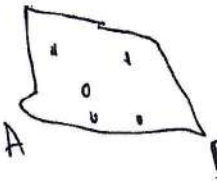
IP адрес	Адрес сети	маска	диапазон	размер	
172.22.228.145/18	172.22.192.0	255.255.192.0	~~255.255.192.1 – 255.255.255.254~~	16382	172.22.255.255
172.22.228.145/21	172.22.224.0	255.255.248.0	172.22.224.1 – 172.22.231.254	2046	172.22.231.245
172.22.228.145/23	172.22.228.0	255.255.254.0	172.22.228.1 – 172.22.229.254	510	172.22.229.255
172.22.228.145/23	172.22.228.128	255.255.255.128	172.22.228.129 – 172.22.228.254	126	172.22.228.255
172.22.228.145/29	172.22.228.144	255.255.255.248	172.22.228.145 – 172.22.228.150	6	172.22.228.151
72.22.228.145/30	172.22.228.144	255.255.255.252	172.22.228.145 – 172.22.228.146	2	172.22.228.147

✓ 172.22.192.1 –
172.22.255.254 ✗

126 содержат ошибки

№ 6

Посмотрим расположение точек, нужных нам. Возьмём произвольную фигуру рассмотрим ~~вокруг~~ сторону BA т.к наши точки многоугольника идут по часовой то ; все точки,



которые нам нужны находятся правее. Если мы для каждой точки проверим, правее ли она всех сторон многоугольника, то мы можем узнать, подходит ли эта точка или нет.

Вариант № 3

Проверить лежит ли сторона правее можно с помощью векторного произведения $x_1 y_2 - x_2 y_1 = |a| |b| \sin\angle(cP)$. Если оно будет <0, то точка лежит левее, если $=0$, то лежит на прямой, если >0, то лежит правее.

Код:

```
# include <bits/stdc++.h>
using namespace std;
struct Point {          // наша точка
 int x, y;
};
Point m( Point a, Point b) {  // вычитание векторов
    Point rez;
    rez.x = a.x - b.x;
    rez.y = a.y - b.y;
    return rez;
}
int cP( Point a, Point b, Point c) {  // векторное произведение
    a = m(a, c);
    b = m(b, c);
   return a.x * b.y - a.y * b.x;
}
signed main() {
   int n; cin >> n;
   vector <Point> v(n); // все точки углов
```

Вариант № 3

```
for(int i=0; i<n; i
int a = -1e9, b = 1
for(int i=0; i<n; i++){
  cin >> v[i].x >> v[i].y;
} k = n
int a = -1e9; // минимальная точка
              // максимальная точка
int b = 1e9;
for (int i=0; i<n; i++) {
 cin >> v[i].x >> v[i].y;
   a = max (v[i].x, max(v[i].y, a));
   b = min (v[i].x, min (v[i].y, a));
}   int an
for (int x=a; x<=b; x++) {
   fo
int ans=0;
for( int x=a; x<=b; x++) { // перебор всех x
   for (int y=a; y<=b; y++) { // перебор всех y
      for(int bool flg=true;
         for (int i=0; i<n; i++) {
            if ( cp(v[i], {x, y}, v[i-1]) <= 0)
                flg = false; // проверка находится ли точка левее отрезка
         }
      if(cp(v[0], {x, y}, v[n-1]) <=0
          flg = false;
      ans = ans + flg; // подсчёт кол-ва
   }
}
cout << ans;
}
```

26 в коде использ. встроенные операции min() max()