Олимпиада школьников «Гранит науки»

БЛАНК ОЛИМПИАДНОЙ РАБОТЫ

Шифр работы 423-01
(не заполнять)

Вариант № 1

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

№ задания		1	2	3	4	5	6	7	8	9	10	Σ
Полученный балл		–	–	0	–	25	20	–	–	–	–	45
I проверка	Подпись проверяющего				Σ баллов прописью	сорок пять						
	Подпись проверяющего				Σ баллов прописью	сорок пять						
II проверка	Подпись проверяющего				Σ баллов прописью	сорок пять						
	Подпись проверяющего				Σ баллов прописью							

Вариант № 1

Задание 5

Будем действовать так:
1. Считаем карту
2. Кластеризуем острова (отделим воду, остров 1 и остров 2)
3. Будем искать остров 2 от острова 1 поиском в ширину.

25

```
class Main:
    def __init__(self):
        self.ma = [[None for _ in range(30)] for _ in range(15)]
        self.a = []
        self.b = []
    def read(self):
        for line in range(15):
            s = input()
            ln = []
            for e in s:
                if e == "~":
                    ln.append(0)
                else:
                    ln.append(1)
            self.ma[line] = ln
    def claster(self):
        ma = self.ma
        # поиск кластера
        for k in [2,3]:
            x,y = -1,-1
            for line in range(15):
                if 1 in ma[line]:
                    for e in range(30):
                        if ma[line][e] == 1:
                            x,y = line,e
            # запись кластера
            queue = [(x,y)]
            old = []
            while queue:
                e = queue.pop(0)
                x,y = e
                if e in old:
                    continue
                if x<0 or x>14 or y<0 or y>29:
                    continue
                if ma[x][y] == 1:
                    ma[x][y] = k
                if k == 2:
                    self.a.append((x,y))
                else:
                    self.b.append((x,y))
                queue.append((x-1,y))
                queue.append((x+1,y))
                queue.append((x,y-1))
                queue.append((x,y+1))
            old.append(e)
    def rename(self):
        d = {0:0, 2:"A", 3:"B"}
        for r in range(15):
            for c in range(30):
                self.ma[r][c] = d[self.ma[r][c]]
    def bridge(self):
        if a == [] or b == []:
            return 0
        br = self.a
        cnt = 1
        while True:
            ne = []
            for e in br:
                x,y = e
                for a,b in [(0,-1),(0,1),(-1,0),(1,0)]:
                    x1,y1 = x+a, y+b
                    if x1<0 or x1>14 or y1<0 or y1>29:
                        continue
                    if self.ma[x1][y1] == 0:
                        ne.append((x1,y1))
                        self.ma[x1][y1] = cnt
                    if self.ma[x1][y1] == "B":
                        return cnt
            br = br + ne
            if ne == []:
                return 0
```

Вариант № 1

```
def run(self):
    self.read()
    self.cluster()
    self.rename()
    return self.bridge()

m = Main()
print(m.run())
```

Вариант № 1

Задание №6

Будем перебирать позиции робота и направления.
Если робот в процессе работы оказывается в той же позиции и с тем же направлением, тогда робот прошёл круг и зациклился.

```
import sys # встроенный модуль для считывания из консоли
class Main:
    def __init__(self):
        self.ma = []
        self.h, self.w = 0,0
    def read1(self):
        for line in sys.stdin:
            ln = []
            for e in line:
                if e == "#":
                    ln.append(0)
                elif e == "~":
                    ln.append(1)
            self.ma.append(ln)
        self.w = len(ln)
        self.h = len(self.ma)

    def read2(self):
        n, m = map(int, input().split())
        for _ in range(m):
            line = input()
            ln = []
            for e in line:
                if e == "#":
                    ln.append(0)
                elif e == "~":
                    ln.append(1)
            self.ma.append(ln)
        self.w = n
        self.h = m

    def try(x,y,n):
        if self.ma[x][y] == 0:
            return 0
        old = []
        old_s = []
        di = { 1: (0,1),
               2: (1,0),
               3: (0,-1),
               0: (-1,0),
             }
        s = 0
        while True:
            if (x,y,n) in old:
                break
            for _ in range(4):
                a,b = di[n]
                if self.ma[x+a][y+b]:
                    old.append((x,y,n))
                    old_s.append((x,y))
                    x,y,n = x+a, y+a, n
                    break
            n = (n+1) % 4
        return len(set(old_s))
    def p(self):
        s = 0
        for x in range(self.h):
            for y in range(self.w):
                for n in range(4):
                    s1 = self.try(x,y,n)
                    s = max(s, s1)
        return s
```

20

205

Вариант № 1

```
def run(self):
    self.read1()
# Если при вводе не указываются размеры квартиры, а только планировка
    self.read2()
# Если указываются размеры и планировка. Тогда модуль sys импортировать
# необязательно
# Использовать только либо read1 либо read2, в зависимости от условий задачи
r=self.p()
return r

r = Main()
print(m.run())
```

Вариант № 1

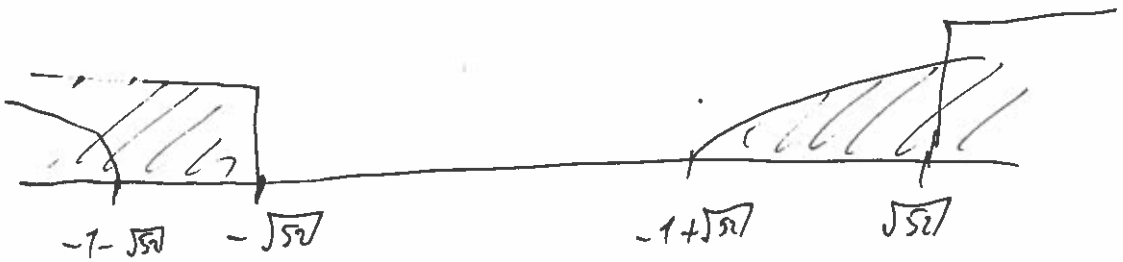
Задание № 3

$A \to B \Longleftrightarrow A \vee \bar{B}$

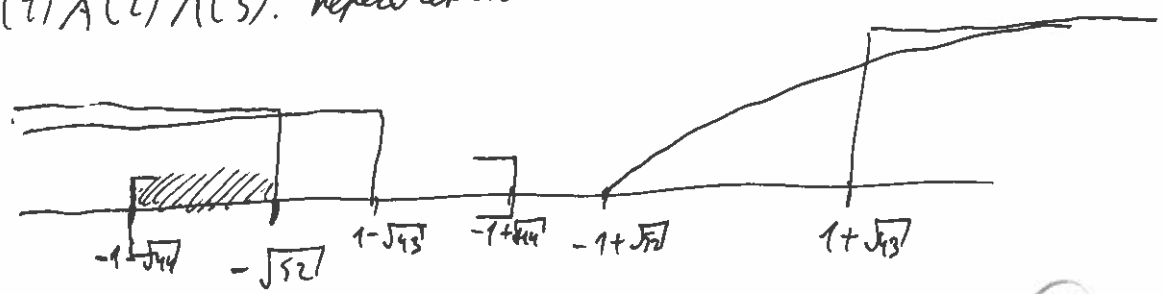
$$\begin{cases} \left[\begin{array}{l} 52 \le x^2 \\ 52 < x^2 - 2x + 1 \end{array}\right. \\ 43 \ge x(x+2) \\ 43 \le (x-1)^2 \end{cases} \qquad \begin{cases} \left[\begin{array}{l} (x-\sqrt{52})(x+\sqrt{52}) \ge 0 \\ (x+1-\sqrt{52})(x+1+\sqrt{52}) > 0 \end{array}\right. (1) \\ (x+1-\sqrt{44})(x+1+\sqrt{44}) \le 0 \ (2) \\ (x-1-\sqrt{43})(x-1+\sqrt{43}) \ge 0 \ (3) \end{cases}$$

$\sqrt{52} \approx 7.2$
$\sqrt{44} \approx 6.6$
$\sqrt{43} \approx 6.5$

(1) - объединение двух решений



(1) ∧ (2) ∧ (3): пересечение областей



$$-1-\sqrt{44} < x < -\sqrt{52}$$

$$-7.6 < x < -7.2$$

Ответ: ~~[illegible]~~ $-\sqrt{52}$

0

Ответ - целое число!
нет мат. решения и таблиц истинности