Олимпиада школьников «Гранит науки» Шифр работы 49-26
БЛАНК ОЛИМПИАДНОЙ РАБОТЫ (не заполнять)

Вариант № 4

Оценка выполнения олимпиадной работы
(заполняется проверяющим)

№ задания		1	2	3	4	5	6	7	8	9	10	Σ
Полученный балл		6	–	–	–	~~10~~ 20	10					~~26~~ 36
I проверка	Подпись проверяющего				Σ баллов прописью	двадцать шесть						
	Подпись проверяющего				Σ баллов прописью	двадцать шесть						
II проверка	Подпись проверяющего				Σ баллов прописью	тридцать шесть						
	Подпись проверяющего				Σ баллов прописью							

№1

1. Из ячейки C5 следует, что в коде нет чисел меньших наибольшего 9.
2. Из ячейки C6 следует, что в коде 4 числовых значения.
3. Из ячейки C8 следует, что E2 = F2, т.к. в ячейке B7 ошибка деления на 0, при этом E2 и F2 – числовые значения.
4. Из ячейки C9 следует, что либо D2 – ~~числовое~~ не числовое значение, либо E2 не числовое значение, либо F2 не числовое значение, т.к. в ячейке B8 ошибка, связанная со складыванием числового и нечислового значения, следовательно D2 – нечисловое значение
5. Из ячейки C10 следует, что F2 = 1, следовательно E2 = 1.
7. Из ячейки C11 следует, что D2 = "e", т.к. D2 – единственное нечисловое значение, буква e в слове гипнотизер находится на 9 месте, длина D2 равна 1, из ячейки C7.
8. Из ячейки C12 следует, что B2 = 2, т.к. C2 ~~нечисловое~~ числовое значение, D2 – нечисловое, длина D2 = 1
6. Из ячейки C7 следует, что длина D2 равна 1
9. Из ячейки C13 следует, что C2 = 10 – B2 – E2 – F2 = 6

Вариант № 4

	B2	C2	D2	E2	F2
Ответ: код:	2	6	e	1	1
	—	—	+	+	+

№ 5. 100

Считаем карту в двумерный массив, где 0 будет обозначать воду, 1 - сушу, потом, с помощью поиска в глубину, найдём клетки, принадлежащие одному из островов, их вместо 1 обозначим цифрой 2. Пройдёмся по полученной карте и считаем координаты клеток, принадлежащих разным островам, в разные массивы. Пройдёмся по одному из массивов. Те клетки, которые с снизу, слева, справа и сверху окружены сушей будем игнорировать. Из них не может вести кратчайший маршрут к другому острову. Остальные клетки будем рассматривать так: ~~будем считать кол-во клеток, которые нужно пройти чтобы попа~~

для каждой клетки первого острова будем пробегаться по каждой клетке второго острова и считать расстояние между ними по их координатам.

Программа: Python 3

```
def dfs(x, y):
    mp[x][y] = 2
    if (mp[x-1][y] == 1):
        dfs(x-1, y)
    if (mp[x+1][y] ==
```

Вариант № 4

Программа Python 3:

```
def dfs(x,y):                    # поиск в глубину для определения клеток
    mp[x][y]=2                   # тем одного из островов
    if x-1>=0:                   # mp - массив, в котором содержится карта
        if mp[x-1][y]==1:
            dfs(x-1,y)
    if x+1<=len(mp)-1:
        if mp[x+1][y]==1:
            dfs(x+1,y)
    if y-1>=0:
        if mp[x][y-1]==1:
            dfs(x,y-1)
    if y+1<=len(mp[x])-1:
        if mp[x][y+1]==1:
            dfs(x,y+1)
#конец поиска в глубину

mp=[]
for i in range(15):              # считывание карты
    mp.append([])
    s=input()
    for j in s:
        if j=="~":
            mp[i].append(0)
        else:
            mp[i].append(1)
```

Вариант № 4

```
b=0                                  len(mp)
for i in range(~~15~~):                  # нахождение первой клетки
    for j in range(len(mp[i])):          # одного из островов
        if mp[i][j]==1:                  # запуск поиска в глубину
            b=1                          # из этой клетки
            dfs(i,j)
        if b:
            break
    if b:
        break
```

20

достойное решение.

```
f=[]    # первый и второй острова
s=[]
                          len(mp)
for i in range(~~15~~):
    for j in range(len(mp[i])):
        if mp[i][j]==1:
            f.append[i,j]
        if mp[i][j]==2:
            s.append[i,j]
ans=1000000000 # в переменной ans будет лежать минимальное
                                                расстояние
~~for i in range(len(f)):~~
    ~~if mp[f[i][0]]~~
    ~~for j~~
```

```
for i in f:
    for j in s:                      # находит минимальное расстояние
        ans=min(ans, abs(i[0]-j[0])+abs(i[1]-j[1]))  # между расстояния от клеток 1-ого
print(ans)                                           # до 2-ого
```

Вариант № 14

№6 106

Будем перебирать все возможные начальные точки запуска робота вместе с направлениями. Для оптимизации не будем ~~зап~~ запускать робота из точек в тех направлениях, в которых ранние были в этой точке при предыдущих запусках.

Программа Python 3:

```
mp=[]                    # mp - массив, в котором описывается комната
for i in range(15):
    s=input()
    mp.append([])
    for j in ~~range(~~s:
        if j=="~":
            mp[i].append(1)
        else:
            mp[i].append(0)

right=[]
left=[]
up=[]
down=[]
for i in range(len(mp)):
    right.append([])
    left.append([])
    up.append([])
    down.append([])
    for j in range(len(mp[i])):
        right.append(0)
        left.append(0)
        up.append(0)
        down.append(0)
```

Вариант № 4

```
ans=0
for i in range(len(mp)):
    for j in range(len(mp[i])):
        t=[]
        for q in range
```

При переборе каждый раз будем создавать 4 массива для всех направлений, в которых может двигаться робот. Массивы будут размером с комнату, изначально будут заполнены 0. При прохождении робота в определённой точке в определённом направлении будем помечать в определённом массиве эту точку как 1. Если робот придёт в точку с единицей, это будет означать, что он начал циклически перемещаться по комнате. На этом моменте прекращаем цикл и переходим к следующей точке для перебора.