Задание №2, Таблицы истинности

Теория

Булевый тип / Логический тип - тип данных в информатике, принимающий два возможных значения, называемых истиной (**true = 1**) и ложью (**false = 0**).

Булевая алгебра / Алгебра логики - раздел математики, занимающийся изучением операций с булевым типом данных.

Таблица истинности - таблица, описывающая логическую функцию, отражающая все значения функции при всех возможных значениях её аргументов.

Операции в алгебре логики (те, что встречаются на ЕГЭ) и слева их обозначения в Python:

¬A, $\overline{A}$,не A — отрицание, инверсия	**not**
A /\ B, A * B, A × B, A и B, A & B, AB — логическое умножение, конъюнкция	**and**
A \/ B, A или B — логическое сложение, дизъюнкция	**or**
A → B — импликация (следование)	**<=**
A ≡ B — эквивалентность (равносильность)	**==**

Таблица истинности булевых операций:

A	B	¬A	A /\ B	A \/ B	A → B	A ≡ B
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Её очень легко усвоить, если запомнить, что:

1. И возвращает 1, только когда оба аргумента равны 1
2. ИЛИ возвращает 0, только когда оба аргумента равны 0
3. Импликация возвращает 0, только когда 1 → 0

> Порядок выполнения логических операций:
>
> 1. Инверсия
> 2. Конъюнкция
> 3. Дизъюнкция
> 4. Импликация
> 5. Эквиваленция

План выполнения 2 задания:

1. Строим таблицу истинности через Python: проходим по всем возможн значениям аргументов (1 и 0 для каждой переменной) через вложенны циклы и выводим в консоль значение функции при данных аргументах.

```
print("x y z w F")
# 1. Перебираем все возможные значения переменных
for x in [0, 1]:
    for y in [0, 1]:
        for z in [0,1]:
            for w in [0,1]:
                # 2. Вычисляем значение функции для каждого набора аргу
                F = ((x or y) and (w <= z)) == (not w)
            # 3. Выводим значения переменных и результат функции в вид
                print(x,y,z,w,int(F))

                        # Если в условии F принимает только 0 или только 1,
                        # можно выводить print только при заданном значени
                        # if F == 0:
                        #     print(x,y,z,w,int(F))
```

2. Сопоставляем полученную таблицу истинности с таблицей из условия: отсеиваем неподходящие строки, смотрим на то, сколько раз переменная принимает то или иное значение, на количество нулей и единиц в строке.

*Если в python "**not**" стоит до или после "**==**" или "**<=**", то необходимо обернуть "**not**" в круглые скобки:*

```
(x == not z) <= not w
```

неправильный вариант

```
(x == (not z)) <= (not w)
```

правильный вариант

Алгоритм решения

Логическая функция F задаётся выражением $(x \vee y \vee z) \rightarrow (x \wedge (y \vee w))$ На рисунке приведён частично заполненный фрагмент таблицы истинности функции F, содержащий **неповторяющиеся строки**. Определите соответствие столбцов таблицы переменным в выражении.

				F
1	0		0	0
	1	1		0
1	1		1	0

В ответе напишите буквы x, y, z, w в том порядке, в котором идут соответствующие им столбцы. Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

1. Выполняем следующий код в Python:

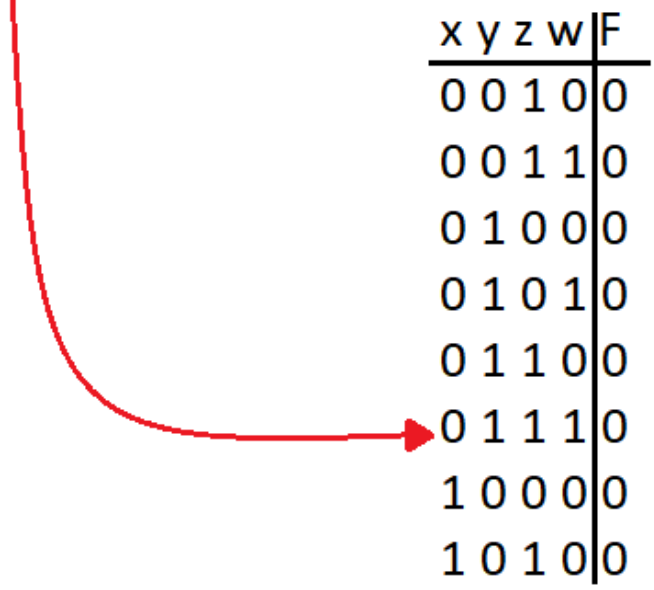
```
print("x y z w F")
for x in [0, 1]:
    for y in [0, 1]:
        for z in [0,1]:
            for w in [0,1]:
                F = (x or y or z) <= (x and (y or w))
                if F == 0: #В таблице из условия F всегда 0, F=1 нас не интересует
                    print (x,y,z,w,int(F))
```

2. Получаем таблицу истинности для F=0:

   x y z w F
   0 0 1 0 0
   0 0 1 1 0
   0 1 0 0 0
   0 1 0 1 0
   0 1 1 0 0
   0 1 1 1 0
   1 0 0 0 0
   1 0 1 0 0

3. Назовём таблицу истинности из условия первой таблицей, а полученную выполнением кода - второй. Замечаем, что последней строке из первой таблицы может соответствовать только строка 0 1 1 1 из второй, т.к. в полученной таблице истинности больше нет строк с 3 и более единицами. Значит, в последней строке первой таблицы пишем 0 на пустое место и понимаем, что этот столбец может соответствовать только переменной **x**

		x		F
1	0		0	0
	1	1		0
1	1	0	1	0



x	y	z	w	F
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	1	0	0

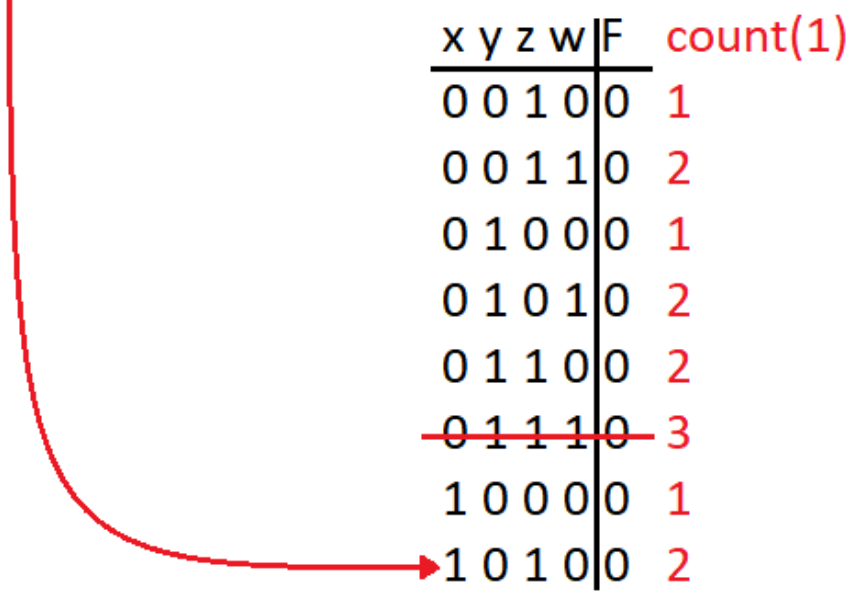
4. Соотнесённые строки больше не несут никакой полезной информации, поэтому можем их игнорировать. Т.к. во второй таблице больше нет строк, где может быть больше 2-х единиц, можно понять, что в пустых ячейках второй строки первой таблицы могут быть только 0.

		x		F
1	0		0	0
0	1	1	0	0
1	1	0	1	0

x y z w	F	count(1)
0 0 1 0	0	1
0 0 1 1	0	2
0 1 0 0	0	1
0 1 0 1	0	2
0 1 1 0	0	2
~~0 1 1 1~~	~~0~~	3
1 0 0 0	0	1
1 0 1 0	0	2

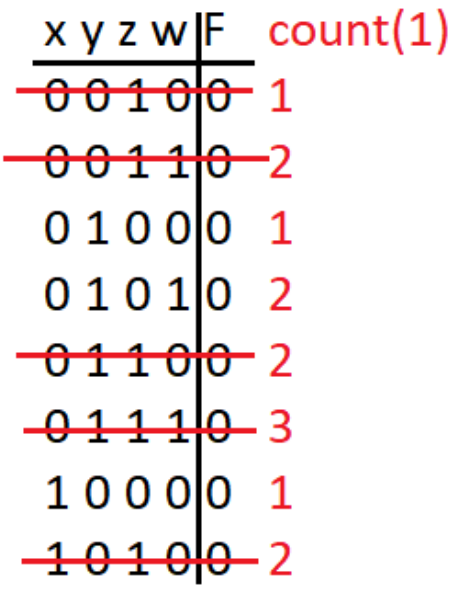
5. **x** во второй строке первой таблицы равен 1. Во второй таблице x = 1 только в двух нижних строках: при 3-ёх нулях и при 2-х. Очевидно, что именно последняя строка второй таблицы соответствует второй строке первой. Вторая единица соответствует **z**, отмечаем.

	z	x		F
1	0		0	0
0	1	1	0	0
1	1	0	1	0



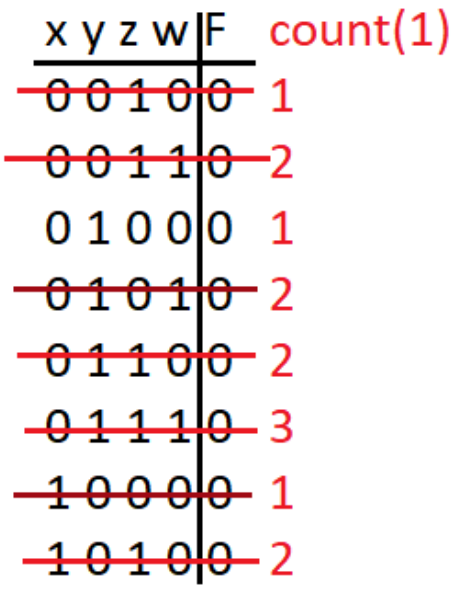
6. Помимо соотнесённой строки, вычеркиваем из второй таблицы все строки, в которых **z**=1, т.к. они точно не соотносятся с оставшейся первой строкой из первой таблицы, в которой **z**=0.

	z	x		F
1	0		0	0
0	1	1	0	0
1	1	0	1	0



7. В оставшейся первой строке первой таблицы в неотмеченных столбцах видим разные значения 1 и 0. Значит, что не соотнесённые **y** и **w** должны во второй таблице так же иметь разные значения. Вычеркиваем неподходящие строки и остаётся лишь одна.

	z	x		F
1	0		0	0
0	1	1	0	0
1	1	0	1	0



8. **x** в оставшейся строке равен 0, **y** = 1, **w** = 0. Подписываем столбцы в первой таблице и получаем ответ: **yzxw**

y	z	x	w	F
1	0	0	0	0
0	1	1	0	0
1	1	0	1	0