23.1) Вася нашел в лесу необычный калькулятор. У этого калькулятора есть всего две команды:

```
1. прибавь 1
2. умножь на 2
```

Первая из них увеличивает заданное число на 1, вторая – умножает на 2. Помогите Васе узнать, сколькими способами можно получить число 26 из числа 5.

Решение: *Решим задание динамически, руками, с помощью таблицы в Excel. Для начала в первой строке таблицы представим все числа от 5 до 26. Во второй строке будет указано кол-во способов, сколькими можно получить число N из числа 5. Для числа 5 это значение будет равно единице.*

5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1																					

Теперь приступим к заполнению таблицы. Делать мы это будем вручную. Для каждого следующего числа N мы будем следовать следующему алгоритму:
1) прибавить кол-во способов, сколькими можно получить число N-1
2) прибавить кол-во способов, сколькими можно получить число N//2 (если N – чётное число)

Например, для чисел 6, 7, 8, 9 – кол-во способов равно 1, т.к. мы можем получить эти числа только прибавляя единицу, т.к. для них (N // 2) нет в таблице.

Для числа 10 кол-во способов = кол-во способов получения 5 + кол-во способов получения 9
Т.е. для 10 N = 1 + 1 = 2

Продолжим заполнение таблицы таким способом до самого конца. Получим следующую таблицу:

5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
1	1	1	1	1	2	2	3	3	4	4	5	5	6	6	8	8	10	10	13	13	16

Т.е. кол-во способов для 26 = 16. Это и есть ответ на задачу.

Ответ: 16

23.2) Исполнитель ПлатиНалоги преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. прибавь 2
2. умножь на 2
3. умножь на 3

Первая из них прибавляет к числу на экране 2, вторая умножает число на экране на 2, третья – умножает число на 3. Сколько существует программ, для которых при исходном числе 1 результатом является число 36 и при этом траектория вычислений не содержит число 20 и содержит число 30?

Решение: *Решим это задание программно, используя рекурсию. Пусть рекурсивная* ***функция f(x, y)*** *возвращает количество программ, для которых при исходном числе* ***x*** *результатом является число* ***y****. Основная идея этой рекурсивной функции в том, что количество программ с результатом* ***y*** *для исходного числа* ***x*** *равно сумме кол-ва программ с результатом* ***y*** *для чисел* ***x+2****,* ***x*2*** *и* ***x*3****.Поэтому, создадим функцию, которая возвращает эту сумму.*

```
def f(x, y):
    return f(x + 2, y) + f(x * 2, y) + f(x * 3, y)
```

Далее, как и у любой рекурсивной функции, у нашей функции должен быть выход из рекурсии. В каком случае мы выходим из рекурсии и что нам надо возвращать? ***Во-первых****, мы выходим из рекурсии, когда получим искомое число* ***y****, т.е. когда* ***x == y****. Это означает, что траектория вычислений привела к заданному числу. Т.е. в данном случае, мы возвращаем 1.*
Во-вторых*, мы выходим из рекурсии, когда мы получим число, большее искомого числа* ***y****. Это означает, что траектория вычислений привела к числу, большему заданному, т.е. траектория вычислений не может привести к результату. В данном случае возвращаем 0.*

```
def f(x, y):
    if x == y: return 1
    if x > y: return 0
    return f(x + 2, y) + f(x * 2, y) + f(x * 3, y)
```

*Рекурсивная функция почти готова. Теперь посмотрим еще раз на условии. В нём сказано: «*при этом траектория вычислений не содержит число 20». *Что это значит? Это значит, что если наша функция привела к числу 20, то мы не рассматриваем, не считаем данную траекторию, т.е. возвращаем 0.*

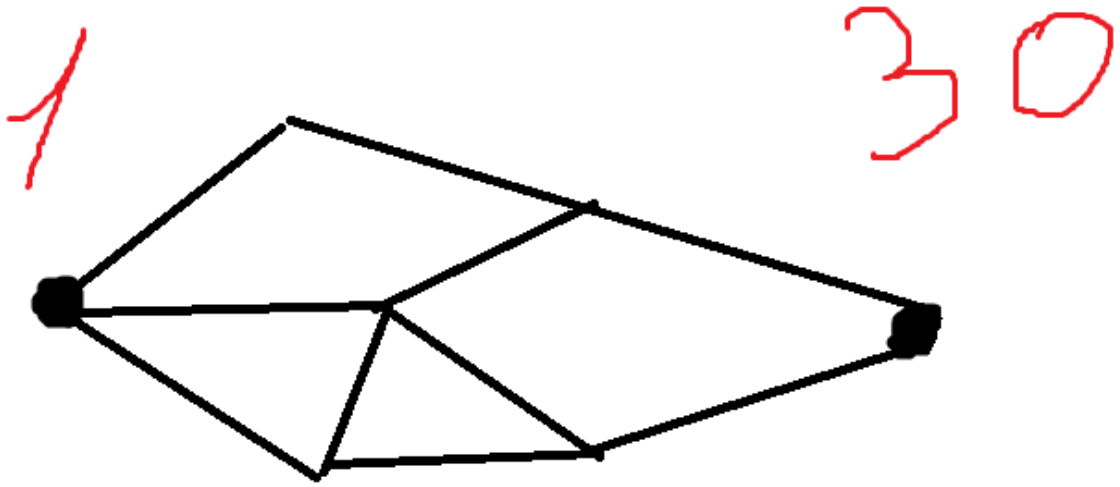
```
def f(x, y):
    if x == y: return 1
    if x > y or x == 20: return 0
    return f(x + 2, y) + f(x * 2, y) + f(x * 3, y)
```

*Рекурсивная функция готова. Теперь используем её в решении. Нам надо найти кол-во «*программ, для которых при исходном числе 1 результатом является число 36*». Т.е.* ***x = 1****,* ***y = 36****. Это будет выглядеть вот таким образом:*

```
print(f(1, 36))
```

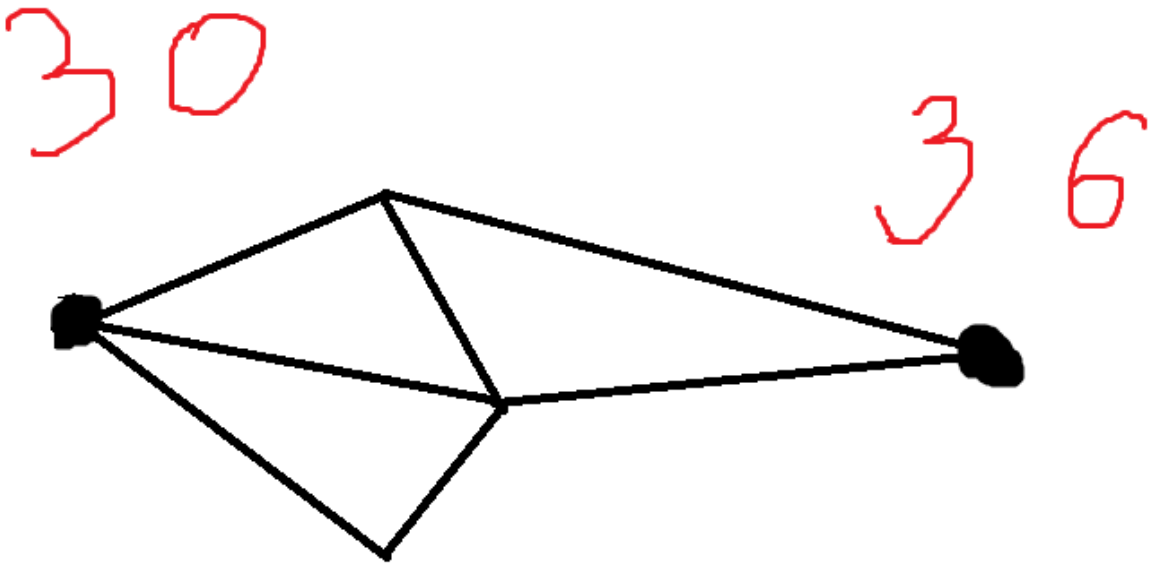
*Но, в условии сказано: «*содержит число 30*». Для этого, нам надо из числа 1 прийти в число 30, затем из числа 30 прийти в число 36, а затем перемножить полученные значения. Почему так*

работает? Давайте обратимся к графам. Схематически, траектории вычислений из 1 в 30 выглядят вот так:

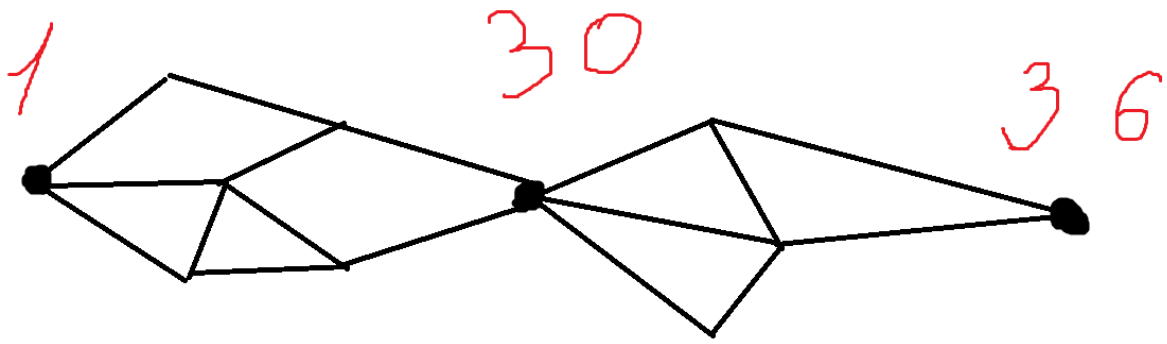


Т.е. на этом графе представлены все (схематически) траектории вычислений, которые ведут из 1 в 30.

Теперь схематически представим траектории вычислений, которые ведут из 30 в 36.



На этом графе представлены схематически все траектории вычислений, которые ведут из 30 в 36. Теперь совместим два графа:



Т.е. вот так выглядят все траектории, для которых из искомого числа 1 результатом является число 36 и при этом траектории содержат число 30. Мы получили два отдельных графа, совместили их. Т.е. мы получили количество для первого, количество для второго графов. Для того, чтобы получить искомое количество, как мы знаем из комбинаторики, нужно перемножить два этих числа.

Именно поэтому мы получаем такую строку в решении:

```
print(f(1, 30) * f(30, 36))
```

Запустив код, мы получим число 64:

```
64
>>>
```

В целом код выглядит очень лаконично и компактно:

```
def f(x, y):
    if x == y: return 1
    if x > y or x == 20: return 0
    return f(x + 2, y) + f(x * 2, y) + f(x * 3, y)

print(f(1, 30) * f(30, 36))
```

Ответ: 64

23.2) Исполнитель Разоритель преобразует число на экране. У исполнителя есть три команды, которым присвоены номера:

1. вычти 2
2. вычти 3
3. раздели на 2

Первая из них вычитает из числа на экране 2, вторая вычитает из числа на экране 3, третья – делит число на экране на 2, если **оно чётное**. Сколько существует программ, для которых при исходном числе 51 результатом является число 3 и при этом траектория вычислений не содержит число 37, содержит число 10 и не имеет двух подряд идущих команд?

Решение: *Решим это задание программно, используя рекурсию. Пусть рекурсивная* ***функция f(x, y, k)*** *возвращает количество программ, для которых при исходном числе* ***x*** *результатом является число* ***y,*** *причём предыдущая команда будет храниться в* ***k****. Для команды* **вычти 2** *будем использовать значок «-2», для* **вычти 3** *- «-3», а для* **раздели на 2** *- «/2». При каждом следующем проходе функции будем проверять, какой была предыдущая команда, а также, будем проверять, чётное ли текущее число* ***x****. В зависимости от этих факторов, от тих критериев, будем возвращать суммы, получаемые разными командами. Также при возврате каждой функции, последним параметром будем указывать команду, которую мы применили, то есть:*

```
f(x - 2, y, "-2")
```

*В данной функции мы к текущему числу применили команду «***вычти 2***», поэтому мы указали её последним параметром* ***k****. Когда эта функция будет выполняться, в ней этот параметр* ***k*** *будем отражать предыдущую команду.*

Выход из рекурсии будет совершать либо когда текущее ***x = y****, в данном случае мы будем возвращать 1; либо когда x < y или x = 37 (т.к. в условии сказано, что траектория вычислений не должна содержать это число), в данном случае мы будем возвращать 0.*

```
def f(x, y, k):
    if x == y: return 1
    if x < y or x == 37: return 0
```

В целом, функция, с обработкой всех критериев, всех факторов, будет выглядеть следующим образом:

```
def f(x, y, k):
    if x == y: return 1
    if x < y or x == 37: return 0
    if k == "-2":
        if x % 2 == 0: return f(x - 3, y, "-3") + f(x // 2, y, "/2")
        else: return f(x - 3, y, "-3")
    if k == "-3":
        if x % 2 == 0: return f(x - 2, y, "-2") + f(x // 2, y, "/2")
        else: return f(x - 2, y, "-2")
    if k == "/2": return f(x - 3, y, "-3") + f(x - 2, y, "-2")
    else:
        if x % 2 == 0: return f(x - 2, y, "-2") + f(x - 3, y, "-3") + f(x // 2, y, "/2")
        else: return f(x - 2, y, "-2") + f(x - 3, y, "-3")
```

В условии сказано, что траектория вычислений должна содержать число 10, поэтому наш вызов функции в основном коде будет выглядеть вот так (также мы должны указать третьим параметром что-то абстрактное, что не подходит под условия в проверке предыдущей команды):

```
print(f(51, 10, "") * f(10, 3, ""))
```

В целом, код выглядит вот так:

```
def f(x, y, k):
    if x == y: return 1
    if x < y or x == 37: return 0
    if k == "-2":
        if x % 2 == 0: return f(x - 3, y, "-3") + f(x // 2, y, "/2")
        else: return f(x - 3, y, "-3")
    if k == "-3":
        if x % 2 == 0: return f(x - 2, y, "-2") + f(x // 2, y, "/2")
        else: return f(x - 2, y, "-2")
    if k == "/2": return f(x - 3, y, "-3") + f(x - 2, y, "-2")
    else:
        if x % 2 == 0: return f(x - 2, y, "-2") + f(x - 3, y, "-3") + f(x // 2, y, "/2")
        else: return f(x - 2, y, "-2") + f(x - 3, y, "-3")

print(f(51, 10, "") * f(10, 3, ""))
```

Запустив его, получим:

```
24
>>>
```

24 – это и есть ответ на задачу.

Ответ: 24