**Задание 2**

//Базовый вариант решения для задачи №2

print('x y w z')  
for x in range(2):  
 for y in range(2):  
 for z in range(2):  
 for w in range(2):  
 if ((y <= (x == w)) and (z <= x)) == False:  
 print(x, y, w, z)

//Упрощенный вариант решения для задачи с использованием модуля itertools

//В функцию task2 вводим данную нам в задаче формулу

from itertools import product  
  
def task2(x, y, w, z):  
 return not(y <= (x == w)) and (z <= x)  
  
print('x y w z')  
  
for (x, y, w, z) in product([0, 1], repeat = 4):  
 if task2(x, y, w, z):  
 print(x, y, w, z)

**Задание 5**

//Программа решения конкретного 5-го задания, реализован вариант с дублированием цифры и добавлением остатка от деления суммы цифр на 2

def R(N):  
 n\_bin = bin(N)[2:]  
 n\_bin = n\_bin + n\_bin[-1]  
 n\_bin += str(n\_bin.count("1") % 2)  
 return int(n\_bin, 2)  
  
for n in range(1, 1000):  
 if R(n) > 97:  
 print(R(n))  
 break

**Задание 6**

//Переборный вариант решения 6-го номера

for i in range(1,10000):  
 s = i  
 n = 0  
 while s <= 375:  
 s = s + 5  
 n = n + 2  
 if n < 295:  
 print(i)

**Задание 7**

//Обрати внимание на комментарии внутри кода

import math  
'''  
N - разрешение экрана (отправлять в функцию тьюплом, либо списком)  
V - объем изображения  
I - количество цветов в палитре изображения  
M - величина объема изображения (0 - БИТ, 3 - БАЙТ, 13 - КБАЙТ, 26 - МБАЙТ)  
'''  
def func(N, V, I, M):  
 if V: # Перевод объема изображения в биты, если объем НЕ в битах  
 V \*= 2\*\*M  
 if N: # Нахождение общего кол-ва точек изображения (?)  
 N = math.prod(N)  
 if N and V and not I: # Если просят найти количество цветов в палитре, также поможет для нахождения кол-во бит на пискель  
 I = 2\*\*(V // N)  
 if I: # Если есть количество цветов в палитре, определит кол-во бит на пиксель (глубину)  
 i = math.ceil(math.log(I) / math.log(2))  
 if N and i and not V: # Если просят найти объем изображения  
 V = N \* i  
 V //= 2\*\*M  
 if V and i and not N: # Если просят найти разрешение изображения  
 N = V // i  
 return N, V, i, I   
'''  
 Пример задания   
Какой минимальный объём памяти (в Кбайт) нужно зарезервировать,   
чтобы можно было сохранить любое растровое изображение размером 64×64 пикселов при условии,   
что в изображении могут использоваться 256 различных цветов?  
Известно: N = (64, 64) / [64, 64] - по условию  
 V = None (т.к. нам надо найти объем изображения)  
 I = 256 - по условию  
 M = 13 - объем просят выразить в КБайтах (смотрим пояснение к аргументу M)  
'''  
print(func((64, 64), None, 256, 13))   
'''  
Получим (4096, 4, 8, 256): за второй параметр отвечает V = 4, записываем в ответ  
'''

**Задание 8(на системы счисления)**

from itertools import product  
cnt = 0  
for i in product("DLORW", repeat=5):  
 cnt += 1  
 if i[0] == "L":  
 print(cnt)

**Задание 8(на любое повторение букв)**

from itertools import product  
cnt = 0  
for i in product("DLORW", repeat=5):  
 cnt += 1  
 if i[0] == "L":  
 print(cnt)

**Задание 8(на перестановки букв)**

from itertools import permutations  
cnt = 0  
for i in permutations("ВИКАЯ", r=4):  
 if "ИЯ" not in "".join(i):  
 cnt += 1  
print(cnt)

**Задание 11**

//Обрати внимание на комментарии внутри кода

import math  
  
'''  
N - длина пароля  
V - объем файла  
A - алфавит пароля (символы из набора...)  
K1 - кодировка пароля (0 - БИТ, 3 - БАЙТ, 13 - КБАЙТ, 26 - МБАЙТ)  
K2 - кодировка объема (0 - БИТ (или оставить всё как было), 3 - БАЙТ, 13 - КБАЙТ, 26 - МБАЙТ)  
D - дополнительная информация  
M - количество пользователей, человек и т.д...  
'''  
  
def func(N, A, V, K1, K2, M, D):  
 i = math.ceil(math.log(A) / math.log(2)) # Узнаем кодировку одного символа  
 V = math.ceil((N \* i) / 2\*\*K1) + D # Узнаем объем по формуле V = (N \* i) / 2\*\*k и добавляем доп. информацию, если есть  
 V \*= M  
 V //= 2\*\*K2 # Превращаем в ту кодировку, которая нужна, если она совпадает с кодировкой на один пароль, то K2 = 0  
 return N, V, i  
  
print(func(15, 13, None, 3, 0, 100, 15))

**Задание 12**

//Решение 12-го задания абсолютно шаблонно, главное - создать первую строку, а далее перезаписать алгоритм из условия

s = "1"\*82  
while "11111" in s or "888" in s:  
 if "11111" in s  
 s = s.replace("11111","88",1)  
 elif "888" in s:  
 s = s.replace("888", "8", 1)  
print(s)

**Задание 14**

//14-ое задание тоже полный шаблон, делим на основание системы счисления, остаток сравниваем с нужной цифрой

s = 5 \*\* 28 + 125 \*\* 100 - 125  
cnt = 0  
while s > 0:  
 if s % 5 == 0:  
 cnt += 1  
 s = s // 5  
print(cnt)

//14-ое задание на двоичную сс – главное забрать число, начиная со второго символа

print(bin(2 \*\* 28 + 64 \*\* 100 - 128)[2:].count("1"))

**Задание 15**

//Здесь собраны все шаблоны для задач №15

def f(x, y, A):  
 return (x - 2\*y < 3\*A) or (2\*y > x) or (3\*x > 50)  
for A in range(1, 1000):  
 if all(f(x, y, A) for x in range(1, 1000) for y in range(1, 1000)):  
 print(A)  
 break

def f(x, A):  
 return (x & 29 != 0) <= ((x & 17 == 0) <= (x & A != 0))  
for A in range(1, 1000):  
 if all(f(x, A) for x in range(1, 1000)):  
 print(A)  
 break  
  
def F(x,A):  
 P=list(range(20,101))  
 Q=list(range(60,141))  
 return ((x in A)<=(x in P)) or (x in Q)

A=list(range(1000))  
for x in range(1000):  
 if not(F(x,A)):  
 A.remove(x)  
print(A)  
print((max(A)-min(A))//10)

**Задание 17**

//По сути меняется лишь 7 строчка с условием

f = open("task17.txt")  
a = []  
sum\_pari = []  
for s in f:  
 a.append(int(s))  
for i in range(len(a)-1):  
 if a[i]\*a[i+1] % 3 == 0:  
 sum\_pari.append(a[i]+ a[i+1])  
print(len(sum\_pari),max(sum\_pari))

**Задание 19-21 на одну кучу**

def WIN1(s):  
 return (s + 1 >= 29) or (s \* 2 >= 29)  
def LOSS1(s):  
 return (not WIN1(s)) and WIN1(s + 1) and WIN1(s \* 2)  
def WIN2(s):  
 return LOSS1(s+1) or LOSS1(s\*2)  
def LOSS12(s):  
 return (WIN1(s + 1) and WIN2(s \* 2)) or (WIN2(s + 1) and WIN1(s \* 2))  
  
s19, s20, s21 = [], [], []  
for s in range(1,29):  
 if LOSS1(s):  
 s19.append(s)  
 if WIN2(s):  
 s20.append(s)  
 if LOSS12:  
 s21.append(s)  
print(s19,s20, s21)

**Задание 19-21 на две кучи**

def WIN1 (x, y):  
 return (x + y + 1 >= win) or (x \* 2 + y >= win) or (x + 2 \* y >= win)  
def LOSS1 (x, y):  
 return (not WIN1(x, y)) and (WIN1(x + 1, y) and WIN1(x, y + 1) and WIN1(x \* 2, y) and WIN1(x, y \* 2))  
def WIN2(x, y):  
 return (LOSS1(x + 1, y) or LOSS1(x, y + 1) or LOSS1(x \* 2, y) or LOSS1(x, y \* 2))  
def LOSS12(x, y):  
 return ((WIN2(x + 1, y) or WIN1(x + 1, y)) and (WIN2(x \* 2, y) or WIN1(x \* 2, y))   
 and (WIN2(x, y + 1) or WIN1(x, y + 1)) and (WIN2(x, y \* 2) or WIN1(x, y \* 2)) and (WIN2(x + 1, y) or WIN2(x \* 2, y) or WIN2(x, y + 1) or WIN2(x, y \* 2))):  
  
x = 16  
s19, s20, s21 = [],[],[]  
for s in range(1, 100):  
 if LOSS1(x, s):  
 s19.append(s)  
 if WIN2(x, s):  
 s20.append(s)  
 if LOSS12(x, s):  
 s21.append(s)  
print(s19, s20, s21)

**Задание 22**

//Переборное решение задания 22

for i in range(1,10000):  
 x = i  
 x0 = x  
 N = 0  
 while x > 0:  
 d = x % 3  
 N = 10 \* N + d  
 x = x // 3  
 N += x0  
 if N >= 100000:  
 print(i)

**Задание 23**

//Вариант решений 23-х номеров, с обычным вариантом, с вариантом обязательных и избегаемых путей, в случае обязательных путей, например, из 1 в 20, через 10, разбиваем один путь на два и результаты перемножаем, как в примере ниже

def task23(start, end):  
 if start == end:  
 return 1  
 if start > end: # сюда так же добавлять запрещенные числа (через которые нельзя проходить), например, if s == 19  
 return 0  
 return task23(start + 1, end) + task23(start \* 2, end)  
print(task23(1, 10) \* task23(10, 20))