Шпаргалка по информатике. Школково.

Задание 2

```
print("x y z w")
for x in range(2):
    for y in range(2):
        for z in range(2):
            for w in range(2):
                if выражение == 0 или 1:
                    print(x, y, z, w)
```

Запись в Python:

∧ - and ¬ - not
∨ - or ≡ - ==
→ - <=

Задание 6

Команды

КУМИР	PYTHON
вперёд(n)	forward(n)
назад(n)	backward(n)
вправо(n)	right(n)
влево(n)	left(n)
опустить хвост	down()
поднять хвост	up()

Пример программы в Python:

```
from turtle import * # подключить библиотеку

k = 10 # коэффициент увеличения
tracer(0) # чтобы моментально вывелось

forward(5 * k) # алгоритм
left(90)
forward(3 * k)
left(45)
for i in range(8):
    forward(6 * k)
    right(90)
up() # поднять перо вверх перед рисованием точек

for x in range(-20 * k, 20 * k, k):
    for y in range(-20 * k, 20 * k, k):
        goto(x, y)
        dot(3, "red")  # точки
done() # чтобы не вышло как нарисовалось
```

Задание 12

```
s = "выражение"
while ".." in s:
# пока ".." в s
if "#" in s:
    s = s.replace('#', '*', 1)
# .replace(old, new, cnt) заменяет
# первое вхождение old на new
print(sum([int(i) for i in s]))
# печатает сумму цифр записи
print(s.count('5'))
# подсчитывает кол-во '5' в строке
```

Задание 7

Перевод:

8 бит = 1 байт
1024 байт = 1 килобайт
1024 килобайт = 1 мегабайт
1024 мегабайт = 1 гигабайт

Формулы:

$N = 2^i$, где N - кол-во цветов, i - глубина кодирования (бит)

Объем изображения **I = x * y * i**, где x - ширина (px), y - высота (px), i - глубина кодирования (биты)

Объем муз. файла **I = f * r * k * t**, где f - частота дикретизации (Гц), r - глубина кодирования, k - кол-во каналов, t - время (сек)

Задание 8

Прототип: "Сколько 4-буквенных слов сможет составить из букв А, Б, В, Г, если при этом А не должна стоять на первом месте и должно быть ровно две буквы?"

```
letters = 'АБВГ'
c = 0
for q in letters:
    for w in letters:
        for e in letters:
            for r in letters:
                word = q + w + e + r
                if word[0] != 'О' and
                        word.count('Б') == 2:
                    c += 1
print(c)
```

Прототип: "Все 4-буквенные слова из букв Г, А, В, Б, записаны в алфавитном порядке и пронумерованы.
Вот начало списка:
1. АААА
2. АААБ
3. АААВ
4. АААГ
5. ААБА
Укажите первое слово, которое не содержит А."

```
letters = 'АБВГ'
c = 0
for q in letters:
    for w in letters:
        for e in letters:
            for r in letters:
                word = q + w + e + r
                c += 1
                if word.count('А') == 0:
                    print(c, word)
```

Задание 15

Неравенства, делимость, побитовая конъюнкция:

```
for A in range(1, 1000):
    flag = 0
    for x in range(1, 1000):
        for y in range(1, 1000): # если есть y
            if выражение == 0: # если по условию должно быть истинно, то ставим 0, иначе 1
                flag = 1
                break
    if flag == 0:
        print(A)
```

Отрезки:

```
def f(x, A):
    P = a1 <= x <= a2
    Q = b1 <= x <= x2
    A = A[0] <= x <= A[1]
    return выражение
    # пример
    # return not(not(A) and P) or Q
borders = [0, 0]
minim = 100000
k = 4
for a in range(0, 80 * k):
    for b in range(a, 80 * k):
        A = [a / k, b / k]
        flag = 0
        for x in range(0, 100 * k):
            if f(x, A) == 0:
                flag = 1
        if flag == 0:
            if A[1] - A[0] <= minim:
                minim = A[1] - A[0]
                borders = a.copy()
print(minim, borders)
```

Множества:

```
def f(x, a):
    P = set([a1, a2, a3, a4])
    Q = set([b1, b2, b3])
    return выражение
    # return ((not(x in a)) <= (x in P)) ...)
# Минимальное кол-во элементов:
a = set()
for x in range(20):
    if f(x, a) == False:
        a.add(x)
print(len(a))
# Максимальное кол-во элементов:
a = set([a1, a2, a3, a4, b1, b2, b3])
for x in range(20):
    if f(x, a) == False:
        a.remove(x)
print(len(a))
```

Задание 14

Перевод из одной СС в другую:

```
# Массив, содержащий в себе цифры
def ss(n, osn):
    res = []
    while n > 0:
        res.append(n % osn)
        n //= osn
    return res[::-1]
```

Операнды арифм. выражения:

```
for x in range(2, 14): # если 14-я СС
    # 5x3
    a = 5 * 14**2 + x * 14**1 + 3 * 14**0
    if a % число == 0:
        print(a, x)
```

Задание 5

Необходимые функции:

bin(n) - перевести из 10 сс в 2 сс
oct(n) - перевести из 10 сс в 8 сс
hex(n) - перевести из 10 сс в 16 сс
int("..", n) - перевести из любой сс в 10 сс

Задание 11

Алгоритм:

1. Считаем мощность алфавита
2. Находим, сколько выделить бит на символ
   $i = \log_2 N$ (округляем вверх)
3. Находим кол-во бит на весь пароль (кол-во символов в пароле умножить на кол-во бит на символ)
4. Переводим в байты (округляем вверх до целого)
5. Считаем кол-во памяти на кол-во людей в байтах
6. Переводим в нужную величину

Задание 23

Прототип: "Кол-во программ из 1 в 28, команды +1, *3, проходит через 5, не проходит через 17".

```
# Динамика
a = [0] * 1000
a[1] = 1
for i in range(1 + 1, 28 + 1):
    if i % 3 == 0:
        a[i] = a[i - 1] + a[i // 3]
    else:
        a[i] = a[i - 1]
    if i == 5:
        for x in range(1, 5):
            a[x] = 0
    if i == 17:
        a[i] = 0
print(a[28])

# Рекурсия
def f(s, e):
    if s == e:
        return 1
    if s > e or s == 17:
        return 0
    return f(s + 1, e) + f(s * 3, e)
print(f(1, 5) * f(5, 28))
```

Задание 19-21

WIN1 - Петя за один ход
LOSE1 - Ваня за один ход (19 задача)
WIN2 - Петя за два хода (20 задача)
LOSE2 - Ваня за два хода (21 задача)

Одна куча

```
from functools import lru_cache
def moves(h):
    return (h+3), (h*2)
@lru_cache(None)
def game(h):
    if h >= 65:
        return 'END'
    elif any(game(x) == 'END' for x in moves(h)):
        return 'WIN1'
# Для 19 на неудачный ход в следующем elif
# не all, a any
    elif all(game(x) == 'WIN1' for x in moves(h)):
        return 'LOSE1'
    elif any(game(x) == 'LOSE1' for x in moves(h)):
        return 'WIN2'
    elif all(game(x) == 'WIN1' or game(x) == 'WIN2'
                                for x in moves(h)):
        return 'LOSE2'
for s in range(1, 63):
    print(s, game(s))
```

Две кучи

```
from functools import lru_cache
def moves(h):
    a, b = h
    return (a + 3, b), (a, b + 3), (a * 2, b), (a, b * 2)
@lru_cache(None)
def game(h):
    if sum(h) >= 65:
        return 'END'
    elif any(game(x) == 'END' for x in moves(h)):
        return 'WIN1'
# Для 19 на неудачный ход в следующем elif
# не all, a any
    elif all(game(x) == 'WIN1' for x in moves(h)):
        return 'LOSE1'
    elif any(game(x) == 'LOSE1' for x in moves(h)):
        return 'WIN2'
    elif all(game(x) == 'WIN1' or game(x) == 'WIN2'
                                for x in moves(h)):
        return 'LOSE2'
for s in range(1, 63):
    h = (5, s)
    print(s, game(h))
```

Задание 16

Если функция превышает глубину рекурсии, то

```
import sys
```

и перед функцией вызвать

```
sys.setrecursionlimit(100000)
```

Задания 3, 9, 18, 22

Полезные формулы:

Найти сумму диапазона	=СУММ(X1:X10)
Найти сумму диапазона с условием	=СУММЕСЛИ(X1:X10; условие)
Найти среднее арифметическое	=СРЗНАЧ(X1:X10)
Найти среднее арифм. с условием	=СРЗНАЧЕСЛИ(X1:X10; условие)
Найти кол-во с условием	=СЧЁТЕСЛИ(X1:X10; условие)
Найти остаток от деления одно на другое	=ОСТАТ(x; y)
Найти значение в другой таблице и вернуть соответствующее ему	=ВПР(искомое значение; диапазон; номер столбца; 0)

Задание 25

```
# Делители числа
def div(x):
    s = set()
    for i in range(1, int(x ** 0.5) + 1):
        s.add(i)
        s.add(x // i)
    return sorted(s)

# Кол-во делителей
def cnt_div(x):
    return len(div(x))

# Определить, простое ли
# число
def is_prime(x):
    if x == 1:
        return False
    for i in range(2, int(x ** 0.5) + 1):
        if x % i == 0:
            return False
    return True
```

Прототип: "Определите числа, не превышающие 10^8, которые соответствуют маске 1*4?25? и делятся на 2025.".

```
from fnmatch import *

for x in range(2025, 10 ** 8, 2025):
    if fnmatch(str(x), "1*4?25?"):
        print(x, x // 2025)
```