Первый (заочный) онлайн-этап научно-образовательного соревнования Олимпиады школьников «Шаг в будущее» по профилю «Инженерное дело» специализации «Профессор Лебедев» (общеобразовательный предмет информатика), осень 2019 г.

11 класс

Вариант 1

Задача 1

Дано натуральное число N < 1 000 000 000. Требуется найти такую цифру в этом числе, у которой модуль разности значения и номера её разряда наибольший.

Разряды числа пронумерованы справа, начиная с единицы.

Ввод: число N.

Вывод: Номер разряда искомой цифры. Если условию удовлетворяет несколько цифр, вывести наибольший номер разряда.

Пример

Пример ввода	Пример вывода
274	2

Проверочные тесты

274	2
9	1
91	2
432	3
14321	5
329	1

Пример решения

```
a = input()
A = list(a)
A.reverse()
ans = 0
index = -1
for i in range(len(A)):
        if (abs(int(A[i]) - (i+1)) >= ans):
        ans = abs(int(A[i]) - (i+1))
        index = i + 1
print(index)
```

Задача 2

Дано натуральное число N < 32768, записанное в десятичной системе счисления. Требуется сформировать "сжатую" запись этого числа в двоичной системе счисления по следующему принципу: вывести номера единичных разрядов этого числа в порядке убывания. Номера разрядов должны быть указаны в шестнадцатеричной системе счисления.
Разряды числа пронумерованы справа, начиная с единицы.
Ввод: число N.
Вывод: "Сжатая" двоичная запись числа N. Цифры шестнадцатеричной системы A..F указывать заглавными буквами.

Пример

Пример ввода	Пример вывода
257	91
2050	C2

Проверочные тесты

257	91
2050	C2
15	4321
32767	FEDCBA987654321
1024	B
1	1

Пример решения

```
a = int(input())
s = []
r = ''
abc = ['A' ,'B' ,'C' ,'D' ,'E' ,'F']
while a >= 1:
        t = a%2
        s.append(t)
        a = a//2
for i in range(0 ,len(s)):
        if s[i] == 1:
        w = i + 1
        if w >= 10:
        g = w - 10
        w = abc[g]
        r = str(w) + r
print(r)
```

Задача 3

Петя придумал свой собственный способ шифрования важной для него информации. Поскольку он не читал ни одной книжки по криптографии, то не знает, что скрывать нужно только ключ шифрования, а строить надёжные шифры на сокрытии алгоритма нельзя. В Петином шифре и данные, и ключ хранятся вместе, весь секрет - в алгоритме.
Петя решил шифровать только тексты на английском языке.
В шифрованной строке подряд записаны заглавные и строчные буквы английского алфавита, пробелы, знаки препинания и цифры. Каждая непрерывная последовательность цифр - число, которое означает, на сколько позиций стоящий слева от него символ будет смещён циклически вправо. При вычислении смещения следует учитывать только символы, не являющиеся цифрами. Если после сдвига символ попадает на последнее место в тексте, он автоматически переносится в начало. Дешифрация осуществляется слева направо.
Ввод: шифрованная строка длиной до 200 символов.
Вывод: расшифрованная строка.

Пример

Пример ввода	Пример вывода
ab1c,2d	a,cbd
etya isP21 cevel11r	Petya is clever

Проверочные тесты

ab1c,2d	a,cbd
etya isP21 cevel11r	Petya is clever
abcd	abcd
aa123aa	aaaa
a5b5c5d5e5f	abcdef
ab1cd2efgh3IJKLMN4OPQRSTU5VW	acbUefdgIJKhLMOPQRNSTVW

Пример решения

```
string = input()
nums = []
len_nums = 0
new_string = ''
chars = []
positions = []
num = ''
for x in range(len(string)):
    try:
    if type(int(string[x])) == int:
    if num == '':
        chars.append(string[x-1])
        positions.append(x-1-len_nums)
    len_nums += 1
    num += string[x]
    except ValueError:
```

```
        new_string += string[x]
        if num != '':
        nums.append(int(num))
        num = ''
        else:
        continue

for i in range(len(nums)):
        tmp_string = new_string[:positions[i]] + new_string[positions[i]+1:]
        l = len(tmp_string)
        pos = positions[i] + nums[i]
        if pos >= l:
        pos = pos%l
        new_string = tmp_string[:pos] + chars[i] + tmp_string[pos:]
print(new_string)
```

Задача 4

Найти в матрице самую длинную последовательность в виде "змейки", в которой все элементы увеличиваются на 1. Каждый последующий элемент последовательности должен располагаться в той же строке или том же столбце, что и предыдущий, на позиции с соседним индексом.

Ввод: первая строка - 2 натуральных числа: размерность матрицы N <= 10 строк и M <= 10 столбцов, записанных через пробел. Далее N строк по M чисел от 0 до 9 в каждой, записанных через пробел.

Вывод: длина искомой последовательности.

Пример

Пример ввода	Пример вывода
2 3 0 3 4 0 0 5	3
3 4 1 3 0 9 4 4 1 0 5 6 7 9	

Проверочные тесты

2 3 0 3 4 0 0 5	3
3 4 1 3 0 9 4 4 1 0 5 6 7 9	4
2 2 1 2 4 3	4
3 3	8

1 2 3 8 0 4 7 6 5	
4 3 2 4 6 4 6 8 1 3 5 3 5 7	1
3 4 1 2 3 4 8 7 6 5 9 0 1 2	9
3 4 0 5 6 1 1 4 7 0 2 3 8 9	10

Пример решения

```
def seek(x, y, w, h, l):
        global longest
        global board
        if x < h - 1 and board[x][y] == board[x + 1][y] - 1:
        seek(x + 1, y, w, h, l + 1)
        if x > 0 and board[x][y] == board[x - 1][y] - 1:
        seek(x - 1, y, w, h, l + 1)
        if y < w - 1 and board[x][y] == board[x][y + 1] - 1:
        seek(x, y + 1, w, h, l + 1)
        if y > 0 and board[x][y] == board[x][y - 1] - 1:
        seek(x, y - 1, w, h, l + 1)
        if l > longest:
        longest = l

board = []
h, w = map(int, input().split())
longest = 0
for i in range(h):
        board.append(list(map(int, input().split())))
for i in range(h):
        for j in range(w):
        seek(i, j, w, h, 1)
print(longest)
```

Задача 5

Многоугольник на плоскости (не обязательно выпуклый) задан координатами своих вершин. Требуется подсчитать количество точек с целочисленными координатами, лежащих внутри него (но не на его границе).

Исходные данные: в первой строке - количество вершин. Далее в каждой строке по 2 целых числа, записанных через пробел - координаты x и y очередной вершины.
Результат: одно целое число - количество точек.

Пример

Пример ввода	Пример вывода
4 1 1 1 3 3 3 3 1	1
8 1 1 1 4 2 4 2 3 3 3 3 4 4 4 4 1	2

Проверочные тесты

4 1 1 1 3 3 3 3 1	1
8 1 1 1 4 2 4 2 3 3 3 3 4 4 4 4 1	2
4 -1 -1 -1 -3 -3 -3 -3 -1	1
3 0 0 1 3 2 0	2
5	4

0 0 1 3 2 1 3 3 4 0	
6 0 0 1 3 2 1 3 3 4 0 2 -1	7

Пример решения

```
def sq(dots):
  s = 0
  for i in range(len(dots) - 1):
        s += dots[i][0] * dots[i + 1][1]
  s += dots[len(dots) - 1][0] * dots[0][1]
  for i in range(len(dots) - 1):
        s -= dots[i + 1][0] * dots[i][1]
  s -= dots[0][0] * dots[len(dots) - 1][1]
  return abs(s) / 2

def nod(a, b):
        while a != 0 and b != 0:
        if a > b:
        a %= b
        else:
        b %= a
        return a + b

num = int(input())
coord = []
for i in range(num):
        coord.append([int(i) for i in input().split()])

dots_on = 0
for i in range(len(coord)):
        dot_1 = coord[i - 1]
        dot_2 = coord[i]
        dots_on += nod(abs(dot_1[0] - dot_2[0]), abs(dot_1[1] - dot_2[1]))
S = sq(coord)

print(int(S - dots_on / 2 + 1))
```

Задача 6

Компьютерные программы на основе "искусственного интеллекта" для игры в шахматы показывают улучшение результатов каждый год. Для этого им требуется быстро и качественно распознавать различные ситуации, возникающие на шахматной доске.

Вам требуется написать программу, которая для заданных "матовых" позиций на доске определит, поставлен ли мат с помощью ладьи или ферзя.

Ввод: в первой строке одно целое число N <= 10 - количество позиций. Далее описывается N шахматных позиций, по 8 строк на каждое описание. В каждой строке через пробел записаны коды фигур, стоящих на поле, либо цифра 0, если в клетке фигуры нет.
Описание фигуры состоит из двух латинских символов: первый - буква w или b, описывающая цвет фигуры - белый или чёрный. Второй символ - код фигуры: король - K, ферзь - Q, ладья - R, конь - N, слон - B, пешка - p.
Гарантируется, что на каждой позиции поставлен мат белым или чёрным.
Вывод: одно число - количество позиций среди заданных, где король находится под шахом ладьи или ферзя.

Пример

Пример ввода	Пример вывода
1 wR 0 0 0 0 0 0 bK 0 0 0 wR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1

Проверочные тесты

1 wR 0 0 0 0 0 0 bK 0 0 0 wR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	1
1 0 0 0 0 0 0 0 bK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wR 0 0 0 0 0 0 wR 0	1
2 wR 0 0 0 0 0 0 bK 0 0 0 wR 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 0 0 0	2

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 bK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wR 0 0 0 0 0 0 0 0	
1 0 0 0 0 0 0 bK 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 wK 0 0 0 0 wR 0 0 0 0 0 0 0 0 0 wQ 0 0 0 0 0 0 wR 0	1
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 bB 0 0 0 0 0 0 0 0 0 0 0 0 bK 0 0 0 0 0 0 0 0 0 0 bN 0 0 0 wK 0 0 0 0 0 0 0	0
1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 bR 0 0 0 0 bB 0 0 0 0 0 0 0 0 0 0 0 0 bK 0 0 0 0 0 0 0 0 0 0 bN 0 0 0 wK 0 0 0 0 0 0 0	0

Пример решения

```
n=int(input())
masf=[]
kol=0
mas_w=[]*5
mas_b=[]*5
for h in range(n):
        mas = [[str(j) for j in (input().split())] for i in range(8)]
        for i in range(8):
        for j in range(8):
```

```
        if mas[i][j]=='bK':
                masf=[]
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_w.append(masf)
        elif mas [i][j]=='wK':
                masf=[]
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_b.append(masf)
        for i in range(8):
        for j in range(8):
        if mas[i][j]=='wR' or mas[i][j]=='wQ':
                masf=[]
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_w.append(masf)
        elif mas[i][j]=='bR' or mas[i][j]=='bQ':
                masf=[]
                masf.append(mas[i][j])
                masf.append(int(i))
                masf.append(int(j))
                mas_b.append(masf)
        if len(mas_w)>1:
        f=False
        pos_k_x=mas_w[0][1]
        pos_k_y=mas_w[0][2]
        for i in range(1,len(mas_w)):
        if not (f) and mas_w[i][1]==pos_k_x or mas_w[i][2]==pos_k_y or
((mas_w[i][0]=='bQ')and(abs(pos_k_x-mas_w[i][1])==abs(pos_k_y-mas_w[i][2]))):
                kol+=1
                f=True
        if len(mas_b)>1:
        f=False
        pos_k_x=mas_b[0][1]
        pos_k_y=mas_b[0][2]
        for i in range(1,len(mas_b)):
        if not (f) and mas_b[i][1]==pos_k_x or mas_b[i][2]==pos_k_y or
((mas_b[i][0]=='wQ')and(abs(pos_k_x-mas_b[i][1])==abs(pos_k_y-mas_b[i][2]))):
                kol+=1
                f=True
        mas_w=[]*5
        mas_b=[]*5
print(kol)
```