Олимпиада школьников «Шаг в будущее»
Заключительный этап

Варинат 1

Задача 1 (5 баллов)

Условие

Для заданного в десятичной системе счисления натурального числа требуется определить, упорядочены ли его цифры в шестнадцатеричном представлении по возрастанию (неубыванию) слева направо.

Входные данные: натуральное десятичное число N, не превышающее 10^6.

Выходные данные: слово YES и через пробел - количество цифр в шестнадцатеричном представлении числа в случае, если цифры упорядочены по неубыванию. В противном случае - слово NO и через пробел - количество пар соседних цифр, для которых условие нарушено.

Пример

Входные данные	Выходные данные	Комментарий
291	YES 3	$291_{10}=123_{16}$
3892	NO 1	$2892_{10}=\mathbf{F3}4_{16}$

Проверочные тесты

Входные данные	Ожидаемый результат
291	YES 3
3892	NO 1
4660	YES 4
773615	YES 5
773630	NO 1
777470	NO 2

Пример решения

```
n = int(input())
c = []
k = 0
t = True
if n == 0:
        c = ['0']
```

```
while n != 0:
        ost = n % 16
        c.append(ost)
        n = n // 16
c = c[::-1]
for i in range(len(c) - 1):
        if c[i] >= c[i + 1]:
        k += 1
        t = False
if t:
        print('YES', len(c))
else:
        print('NO', k)
```

Задача 2 (8 баллов)

Условие

Требуется определить количество переносов между разрядами, которые потребуются при вычислении суммы двух чисел в десятичной системе счисления.
Входные данные: через пробел записаны 2 натуральных десятичных числа, каждое не превышает 10^6.

Выходные данные: одно число - количество переносов, которые потребуются при вычислении суммы чисел в десятичной системе счисления.

Пример

Входные данные	Выходные данные
123 456	0
789 65	2

Проверочные тесты

Входные данные	Ожидаемый результат
123 456	0
789 65	2
1 1	0
55 55	2
9999 1	4
876543 123456	0

Пример решения

```
x,y = map(str, input().split())
k = 0
if x < y:
        x,y = y,x
y = '0'*(len(x)-len(y)) + y
delta = 0
for i in range(len(y)):
        a = int(x[-1-i])
        b = int(y[-1-i])
        if a + b + delta >= 10:
        delta = int(str((a + b + delta))[0])
        k += 1
        else:
        delta = 0

print(k)
```

Задача 3 (10 баллов)

Условие

Самым простым методом сортировки считается сортировка "пузырьком". Алгоритм основывается на том, что каждый шаг сортировки состоит в проходе слева направо по массиву, при этом просматриваются пары соседних элементов. Если элементы некоторой пары находятся в неправильном порядке, то осуществляется обмен их местами.

В этой задаче требуется определить количество перестановок, которые будут выполнены при сортировке некоторого числового массива методом пузырька.

Входные данные: в первой строке записано натуральное число N - количество элементов массива, не превышающее 100. В последующих N строках записаны натуральные элементы массива, каждый не превышает 100.

Выходные данные: количество перестановок элементов, которые потребуются для сортировки массива методом "пузырька" по возрастанию.

Пример

Входные данные	Выходные данные	Примечание
4 1 4 2 3	2	Перестановки: 1. 4 и 2, новый масив - 1 2 4 3 2. 4 и 3, итоговый массив - 1 2 3 4

5 1 4 3 2 5	3	Перестановки: 1. 4 и 3, новый массив - 1 3 4 2 5 2. 4 и 2, новый массив - 1 3 2 4 5 3. 3 и 2, итоговый массив - 1 2 3 4 5

Проверочные тесты

Входные данные	Ожидаемый результат
4 1 4 2 3	2
5 1 4 3 2 5	3
6 6 1 2 3 4 5	5
6 2 3 4 5 6 1	5
5 5 4 3 2 1	10

1 3	0

Пример решения

```
a = int(input())
n = []
for i in range(a):
        n.append(int(input()))
k = 0
for i in range(len(n) - 1):
        for j in range(len(n) - 1):
        if n[j] > n[j + 1]:
        k += 1
        n[j], n[j + 1] = n[j + 1], n[j]
print(k)
```

Задача 4 (12 баллов)

Условие

Для заданного набора точек на плоскости требуется упорядочить их так, чтобы они располагались по часовой стрелке, начиная от вектора, направленного по оси Y.

Входные данные: в первой строке записано натуральное количество точек N, не превышающее 20. В последующих N строках через пробел записаны вещественные координаты X и Y каждой точки, разделителем целой и дробной части служит точка. Значение каждой координаты по модулю не превышает 100, длина дробной части не превышает 3 знаков.

Точек с одинаковыми координатами во входных данных нет.

Выходные данные: номера точек, упорядоченных по часовой стрелке, каждый в отдельной строке. Считать, что нумерация начинается с 1. Если для каких-либо точек угол между осью ординат и отрезком между точкой и началом координат совпадает, то сначала вывести номер точки, расположенной ближе к началу координат.

Пример

Входные данные	Выходные данные
5 3 3 1 5 -3 2 4 1	2 1 4 5 3

-5 -4	
3 -1.1 -3 -10.3 -0.5 -2 -2	1 3 2

Проверочные тесты

Входные данные	Ожидаемый результат
5 3 3 1 5 -3 2 4 1 -5 -4	2 1 4 5 3
3 -1.1 -3 -10.3 -0.5 -2 -2	1 3 2
1 1 1	1
2 0.1 1 -1 -0.1	1 2
4 0 -5 -5 0 0 5 5 0	3 4 1 2
3 -10.08 7 -10.12 3 -10.1 5	2 3 1

Пример решения

```
dots = dict()
```

```
for i in range(int(input())):
        dots[i + 1] = [float(i) for i in input().split()]

def sorting(arg):
        if arg[0] == 0 and arg[1] > 0:
        return (0, arg[1])

        elif arg[0] > 0 and arg[1] > 0:
        return (1, arg[0] / arg[1], arg[0] ** 2 + arg[1] ** 2)
        elif arg[1] == 0 and arg[0] > 0:
        return (1.5, arg[0])

        elif arg[0] > 0 and arg[1] < 0:
        return (2, -arg[1] / arg[0], arg[0] ** 2 + arg[1] ** 2)
        elif arg[0] == 0 and arg[1] < 0:
        return (2.5 , -arg[1])

        elif arg[0] < 0 and arg[1] < 0:
        return (3, arg[0] / arg[1], arg[0] ** 2 + arg[1] ** 2)
        elif arg[1] == 0 and arg[0] < 0:
        return (3.5, -arg[0])

        elif arg[0] < 0 and arg[1] > 0:
        return (4, -arg[1] / arg[0], arg[0] ** 2 + arg[1] ** 2)

for i in sorted(dots.keys(), key=lambda x: sorting(dots[x])):
        print(i)
```

Задача 5 (15 баллов)

Условие

Вася хочет заниматься разработкой компьютерных игр. В ближайших планах у него - создание игры в жанре двухмерной стратегии, для которой необходимо научиться генерировать интересные игровые карты. По задумке Васи, любая карта представляет собой участок суши, через который протекает одна или несколько рек.

Вам требуется помочь Васе для некоторой случайно сгенерированной карты найти самую длинную реку, которая не имеет участков, расположенных левее её истока.

Вася решил, что все реки текут с севера на юг, то есть истоком на карте является тот конец реки, у которого номер строки ниже. Если оба конца реки имеют одинаковый номер строки, то исток - тот, у которого номер столбца меньше.

Реки пересекаться не могут, но могут сливаться или разделяться. После соединения или разделения все продолжения реки могут продолжаться только вниз или вправо.

По границе карты река течь не может.

Разделяться река может не более, чем на два рукава в одной точке, и сливаться в одной точке могут не более двух рукавов реки.

Также река может иметь исток на самой карте, а не брать своё начало за пределами карты.

Входные данные: квадратная матрица из нулей и единиц, на которой 0 означает сушу, а 1 означает реку. Соседними ячейками одной и той же реки являются такие, у которых индексы строки или столбца отличаются на 1.

Выходные данные: координаты начала и конца искомой реки. В первой строке требуется через пробел вывести номера строки и столбца левого верхнего элемента матрицы, являющегося "истоком" реки на карте. Во второй строке требуется через пробел вывести номера строки и столбца, являющегося другим концом реки на карте.

Нумерация строк и столбцов ячеек матрицы начинается с 1.

Пример

Входные данные	Выходные данные
0100010 0110110 0010100 0010110 0010010 0010010 0010010	1 2 7 3
01000000 01000000 01110000 00010000 00011110 00010010 00010010 00010010	1 2 8 7

Проверочные тесты

Входные данные	Ожидаемый результат
0100010 0110110 0010100 0010110	1 2 7 3

Олимпиада школьников «Шаг в будущее»
Заключительный этап

0010010 0010010 0010010	
01000000 01000000 01110000 00010000 00011110 00010010 00010010 00010010	1 2 8 7
010 010 010	1 2 3 2
010100 010100 010110 010010 010010 010010	1 4 6 5
0100000100 0111000100 0001000100 0001000100 0001000111 0001000000 0001000000 0001111111 0000000000 0000000000	1 2 8 10
0001000100 0001000100 0001000100 0011000100 0010000111 0010000000 0010000000 0011111111 0000000000	1 8 5 10

0000000000	
0001000000 0001000000 0001110000 0000010000 0001110000 0001010000 0001010000 0001010000 0001010000 0001010000	1 4 10 4
0001000100 0001000100 0001000100 0001111100 0000000100 0000000100 0000000100 0000000100 0000000100 0000000100	1 4 10 8

Пример решения

```
from sys import stdin

class River:
	def __init__(self, p):
	self.p = p
	self.l = False

	def add_point(self, p):
	used_points.add(p)
	self.p.append(p)

	def check_river(self):
```

```
        for i in self.p[1:]:
        if i[1] < self.p[0][1]:
                self.l = True
                break

def count_river(r):
        now = r.p[-1]
        while True:
        pos_moves = []
        if now[1] > 0:
        if matrix[now[0]][now[1] - 1] == "1" and (now[0], now[1] - 1) not in r.p:
                pos_moves.append((now[0], now[1] - 1))
        if now[0] < size - 1:
        if matrix[now[0] + 1][now[1]] == "1" and (now[0] + 1, now[1]) not in r.p:
                pos_moves.append((now[0] + 1, now[1]))
        if now[1] < size - 1:
        if matrix[now[0]][now[1] + 1] == "1" and (now[0], now[1] + 1) not in r.p:
                pos_moves.append((now[0], now[1] + 1))
        if len(pos_moves) == 0:
        break
        elif len(pos_moves) == 1:
        m = pos_moves[0]
        r.add_point(m)
        now = m
        else:
        for m in pos_moves[:-1]:
                used_points.add(m)
                n_r = River(r.p + [m])
                rivers.append(n_r)
                count_river(n_r)
        m = pos_moves[-1]
        r.add_point(m)
        now = m

rivers = []
used_points = set()
matrix = []

for i in stdin:
        matrix.append(list(i.strip()))
size = len(matrix)
for row in range(len(matrix)):
        for col in range(len(matrix[row])):
        if matrix[row][col] == "1":
        if (row, col) not in used_points:
                used_points.add((row, col))
                r = River([(row, col)])
                rivers.append(r)
                count_river(r)
for r in rivers:
```

```
            r.check_river()
m_r = max(rivers, key=lambda x: (len(x.p) if not x.l else 0))
print(*[i + 1 for i in m_r.p[0]])
print(*[i + 1 for i in m_r.p[-1]])
```