Бушуев Максим Сергеевич

Балл: 44

Задача №1

Условие

Учительница математики Марина Григорьевна попросила старшеклассников составить программу для тестирования младших учеников по геометрии, которая будет выдавать задачи на проверку принадлежности точки - прямой.

Требуется написать программу, которая будет контролировать правильность ответов учеников.

Входные данные: в первой строке уравнение вида ax+by+c=0, где a, b и c - целые числа от 0 до 1000, а знаками операции могут быть как "+", так и "-". Во второй строке - целые числа - координаты проверяемой точки.

Выходные данные: если точка лежит на прямой - слово "YES", иначе - слово "NO" и через пробел - целая часть расстояния от точки до прямой.

Пример

Исходные данные	Результат
1x+1y+0=0 1 -1	YES
2x-3y+1=0 2 0	NO 1

Исходный код

```
import math
code = str(input())
nums = []
act = []
for i in range(len(code)):
    if code[i] == "x":
        nums.append(int(code[0:i]))
        act.append(code[i + 1])
        k = int(i)
    if code[i] == "y":
        nums.append(int(code[k + 2:i]))
        act.append(code[i + 1])
        k = int(i)
    if code[i] == "=":
        nums.append(int(code[k + 2:i]))
        break
chek = input().split()
a = nums[0] * int(chek[0])
b = nums[1] * int(chek[1])
if act[0] == "+":
    a += b
else:
    a -= b
if act[1] == "+":
    a += nums[2]
else:
    a -= nums[2]
if a == 0:
    print("YES")
else:
    x1 = (-nums[1] * int(chek[1]) - nums[2]) / nums[0]
    y1 = int(chek[1])
    y2 = (-nums[0] * int(chek[0]) - nums[2]) / nums[1]
    x2 = int(chek[0])
    x = x1 - x2
    y = y1 - y2
    L = math.sqrt(x ** 2 + y ** 2)
    if x1 * x2 > 0:
        if x1 > 0:
            x = max(x1, x2) - min(x1, x2)
        else:
            x = max(x1, x2) - min(x1, x2)
    elif x1 == 0:
        if x2 > 0:
            x = x2
        else:
            x = -x2
```

```
    elif x2 == 0:
        if x1 > 0:
            x = x1
        else:
            x = -x1
    else:
        if x1 > 0:
            x = x1 - x2
        else:
            x = x2 - x1
    h = int((x * y) / L)
    print("NO", h)
```

Задача №2

Условие

В строке записано истинное логическое выражение с тремя переменными a, b и c. Над переменными применяются 2 операции: эквивалентность и исключающее «или». Требуется восстановить значения таблицы истинности функции, соответствующей этому выражению.

Исключающее «или» - булева функция двух переменных, результат которой истинен тогда и только тогда, когда один аргумент истинен, а второй - ложен.

Эквивалентность - логическое выражение, которое является истинным тогда, когда оба простых логических выражения (левая и правая части) имеют одинаковую истинность.

В рамках данной задачи будем обозначать исключающее «или» знаком "^", а эквивалентность - знаком "=" без кавычек.

Входные данные: логическое выражение, состоящее из имён переменных a, b, c, знаков исключающего «или» и эквивалентности. Длина выражения не превышает 20 символов.

Выходные данные: 8 цифр 0 и 1, записанных неразрывно и означающих значения функции для каждой из комбинаций значений переменных:

0, 0, 0;

0, 0, 1;

0, 1, 0;

0, 1, 1;

1, 0, 0;

1, 0, 1;

1, 1, 0;

1, 1, 1.

Пример

Исходные данные	Результат
a^b^c	01101001
a^b=b^c	10100101

Примечание: операция исключающего «или» имеет более высокий приоритет по сравнению с эквивалентностью.

Исходный код

```
chek = [[0, 0, 0], [0, 0, 1], [0, 1, 0], [0, 1, 1], [1, 0, 0], [1, 0, 1], [1, 1, 0], [1, 1, 1]]
input_str = input()
answ = ''

for val in chek:
    strok = ''
    for i in input_str:
        if i == "a":
            i = str(val[0])
        elif i == "b":
            i = str(val[1])
        elif i == "c":
            i = str(val[2])
        strok += str(i)
    strok = strok.split('=')
    if len(strok) != 1:
        for i in range(len(strok)):
            a = []
            for k in strok[i]:
```

```
                a.append(k)
            while len(a) != 1:
                if a[0] == a[2]:
                    a[0] = "0"
                else:
                    a[0] = "1"
                del a[1:3]
            strok[i] = str(a[0])
        while len(strok) != 1:
            if strok[0] == strok[1]:
                strok[0] = "1"
            else:
                strok[0] = "0"
            del strok[1]
        answ += str(strok[0])
    else:
        n = []
        for a in strok[0]:
            n.append(a)
        while len(n) != 1:
            if n[0] == n[2]:
                n[0] = "0"
            else:
                n[0] = "1"
            del n[1:3]
        answ += str(n[0])
print(answ)
```

Задача №3

Условие

Дана запись двух больших целых чисел в шестнадцатеричной системе счисления и их произведения, при этом в записи допущена одна ошибка.

Написать программу, которая выявит допущенную ошибку и исправит её.

Входные данные: три строки, в каждой из которых записано по шестнадцатеричному числу. Количество знаков в каждом из чисел не превышает 100. Цифры от A до F записаны заглавными латинскими буквами.

Выходные данные: исправленная запись в том же формате.

Пример:

Исходные данные	Результат
10 A A2	10 A A0
1F 27 49A	1F 26 49A

Исходный код

```
a = input()
b = input()
c = input()
raz = []
def abc(a):
    if a == "A":
        return 10
    elif a == "B":
        return 11
    elif a == "C":
        return 12
    elif a == "D":
        return 13
    elif a == "E":
        return 14
    elif a == "F":
        return 15
    else:
        return int(a)
def to16(a):
    n = 0
    for i in range(len(a) - 1, -1, -1):
        n += abc(a[i]) * 16 ** (len(a) - i - 1)
    return n
x = to16(a)
y = to16(b)
z = to16(c)
```

```
raz.append(z / x)
raz.append(z / y)
if min(raz) % 1 == 0:
    if min(raz) == raz[0]:
        y = int(raz[0])
    else:
        x = int(raz[1])
elif max(raz) % 1 == 0:
    if max(raz) == raz[0]:
        y = int(raz[0])
    else:
        x = int(raz[1])
else:
    z = x * y
def cba(a):
    if a == 10:
        return "A"
    elif a == 11:
        return "B"
    elif a == 12:
        return "C"
    elif a == 13:
        return "D"
    elif a == 14:
        return "E"
    elif a == 15:
        return "F"
    else:
        return str(a)

def from16(a):
    c = ''
    e = ""
    while a != 0:
        b = a - (a // 16) * 16
        a = a // 16
        c += cba(b)
    for i in range(len(c) - 1, -1, -1):
        e += c[i]
    return e
print(from16(x))
print(from16(y))
print(from16(z))
```

Задача №4

Условие

Пете нравится решать судоку, и он задумался, как можно изменить эту головоловку, чтобы она стала интереснее. После размышлений ему пришла идея сделать судоку в 16-ричной системе счисления, но он засомневался, получится ли её решать так же, как обычную.

Правила 16-ричных судоку, которые придумал Петя: дано поле 16x16, разделённое на квадраты 4x4. Допустимы все возможные 16-ричные цифры от 0 до F, при этом не должно быть повторов одной цифры в строках и столбцах поля, а также в отдельных квадратах.

Требуется написать программу, которая заполнит недостающие клетки в заданном 16-ричном судоку.

Входные данные: 16 строк по 16 символов с цифрами от 0 до F. Пустые поля обозначены символами "_" (знак подчёркивания). Количество пустых полей не превышает 30.

Выходные данные: решённое судоку - 16 строк по 16 символов с цифрами от 0 до F, такие, что все пустые поля заполнены пропущенными цифрами.

Пример:

Исходные данные	Результат

12_456_890ABCDEF	1234567890ABCDEF
567890ABCDEF1234	567890ABCDEF1234
90ABCDEF12345678	90ABCDEF12345678
CDEF123_567890AB	CDEF1234567890AB
3_567890A_CDE_12	34567890ABCDEF12
7890ABCDEF123456	7890ABCDEF123456
ABCDEF1234567890	ABCDEF1234567890
EF12_4567890ABCD	EF1234567890ABCD
678_0ABCDEF12345	67890ABCDEF12345
234__7890A__DEF1	234567890ABCDEF1
DEF1234567890_BC	DEF1234567890ABC
0ABC_EF123456789	0ABCDEF123456789
890ABCDEF123456_	890ABCDEF1234567
_567890ABCDEF1_3	4567890ABCDEF123
_1234567890ABCDE	F1234567890ABCDE
BCDEF123456789_A	BCDEF1234567890A

Примечание: гарантируется, что исходные данные корректны, судоку может быть решено и решение единственное.

Исходный код

Задача №5

Условие

Двоичным деревом называется иерархическая структура данных, в которой каждый узел имеет не более двух потомков.

Двоичное дерево поиска - разновидность двоичного дерева, у которого оба поддерева (левое и правое) являются двоичными деревьями поиска; все узлы левого поддерева произвольного узла A меньше, чем значение самого узла A; все узлы правого поддерева произвольного узла A больше либо равны значению узла A.

Высотой узла называется максимальная длина нисходящего пути от этого узла к самому нижнему узлу, называемому листом. Высота корневого узла равна **высоте** всего **дерева**.

Дерево называется **сбалансированным**, если для любой его вершины высота левого и правого поддерева для этой вершины различаются не более чем на 1.

При **прямом обходе** дерева сначала обрабатывается ключ (значение) корня, затем - ключи левого и правого поддеревьев.

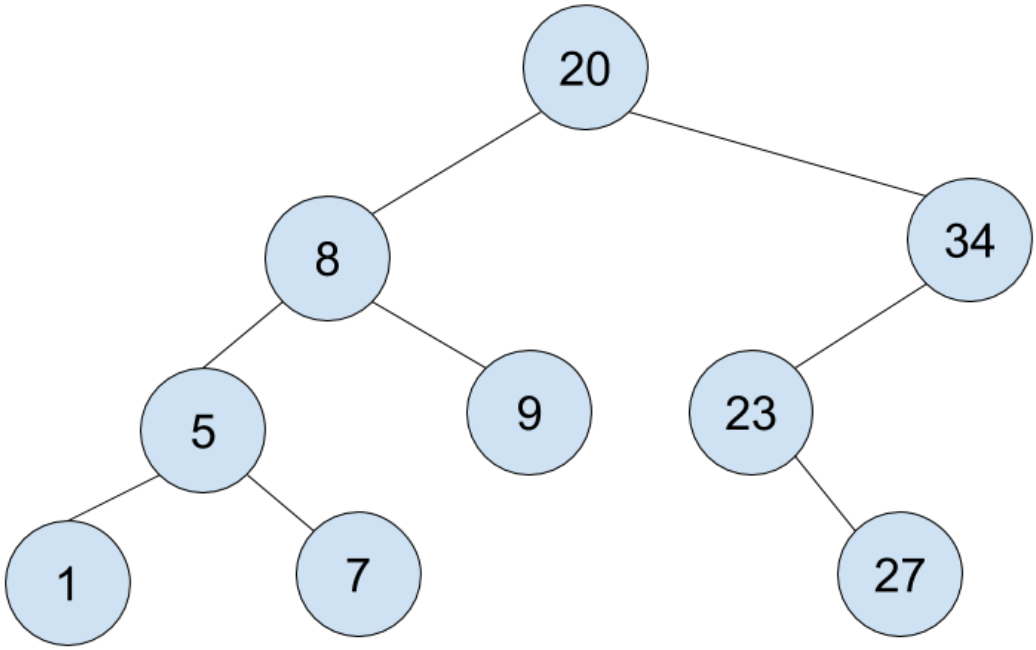
По заданным ключам вершин, полученным при прямом обходе, требуется определить, является ли дерево сбалансированным, а также определит высоту дерева.

Входные данные: ключи всех вершин двоичного дерева поиска в порядке прямого обхода. Каждый ключ - натуральное число, не превышающее 10_5. Каждое значение задано в отдельной строке. Последняя строка состоит из одного символа - точки.

Выходные данные: первая строка - слово YES, если дерево сбалансированное, и NO, если нет. Вторая строка - число, соответствующее высоте дерева.

Пример

Исходные данные и результат соответствуют дереву, изображённому на рисунке:



Исходные данные	Результат
20 8 5 1 7 9 34 23 27 .	NO 4

Исходный код