

Министерство цифрового развития связи и массовых коммуникаций РФ

Государственное бюджетное образовательное учреждение высшего  
образования

Ордена Трудового Красного Знамени

«Московский технический университет связи и информатики»

Кафедра «Математическая кибернетика и информационные технологии»

дисциплина «Структуры и алгоритмы обработки данных»

Отчет по курсовой работе

Подготовил: студент группы

БВТ1903 Саввин Д.И.

Проверил: Кутейников И.А.

Москва

2021

## Оглавление

1. ЦЕЛЬ РАБОТЫ .....	3
1.nba .....	3
2.parliament.....	3
3.strings .....	4
4.problems .....	4
2. ВЫПОЛНЕНИЕ .....	6
1.nba .....	6
2.parliament.....	10
3.strings .....	15
4.problems .....	19
3. ВЫВОД .....	25

## 1. ЦЕЛЬ РАБОТЫ

Применить на практике навыки, полученные в ходе выполнения лабораторных работ, для решения различных задач.

Задания:

1.nba

1) Вам заданы параметры некоторых из 32 игроков текущего драфта (для объективности имена игроков не указаны). Ваша задача – для каждого игрока определить категорию, под которую он подпадает для вашей команды на драфте.

2.parliament

1) Вам дана запись о событиях на сессии парламента. Каждое событие является либо внесением нового законопроекта, либо голосованием за какой-то законопроект, причём в обоих случаях известен номер партии, этот законопроект предложившей. События даны в том порядке, в котором они происходили. Проверьте, может ли данная запись о событиях соответствовать какому-либо заседанию, удовлетворяющему порядку проведения, описанному выше.

2) Задан правильный  $N$ -угольник. Требуется выбрать наименьшее количество его вершин, которые также образуют правильный многоугольник.

3) На форуме, на котором обсуждаются задачи олимпиад по информатике, ввели следующий аналог капчи. Участнику выдаётся строка из  $N$  десятичных цифр (без ведущих нулей). В качестве ответа требуется ввести такое основание системы счисления  $B$ , что в этой системе счисления выданная запись будет соответствовать составному

числу (назовем его  $D$ ), а также число  $X$ , большее 1 и меньшее  $D$ , являющееся делителем  $D$ .

При этом  $B$  и  $X$  не должны превосходить 109.

По заданной строке десятичных цифр найдите любую пару чисел  $B$  и  $X$ , удовлетворяющую ограничениям, или ответьте, что решения в заданных ограничениях не существует.

### 3.strings

1) Даны две строки:  $s1$  и  $s2$  с одинаковым размером, проверьте, может ли некоторая перестановка строки  $s1$  “победить” некоторую перестановку строки  $s2$  или наоборот. Строка  $x$  может “победить” строку  $y$  (обе имеют размер  $n$ ), если  $x[i] \geq y[i]$  (в алфавитном порядке) для всех  $i$  от 0 до  $n-1$ .

2) Дана строка  $s$ , вернуть самую длинную полиндромную подстроку в  $s$ .

3) Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как  $a + a$ , где  $a$  - некоторая строка).

### 4.problems

1) Массив  $A$  состоит из целых положительных чисел длин отрезков. Составьте из трех отрезков такой треугольник, чтобы его периметр был максимально возможным. Если невозможно составить треугольник с положительной площадью функция возвращает 0.

2) Дан массив неотрицательных целых чисел  $nums$ . Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

3) Дана матрица `mat` размером  $m * n$ , значения целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу.

## 2. ВЫПОЛНЕНИЕ

1.nba

Рассмотрим решение задачи с драфтом игроков NBA. Ниже представлен класс описывающий отдельного игрока.

player.py

```
class player:

    def __init__(self, height, wingspan, avg_points, avg_takes, avg_passes):
        self.height = height
        self.wingspan = wingspan
        self.avg_points = avg_points
        self.avg_takes = avg_takes
        self.avg_passes = avg_passes

    def get_player_category(self):
        top_half = 0
        count_goods = 0
        count_sgoods = 0
        green_unicorn = False
        # Проверка роста
        if (self.height >= 190) and (self.height <= 220):
            count_goods += 1
            if self.height > 205:
                top_half += 1
        elif self.height > 220:
            count_sgoods += 1
            green_unicorn = True
        # Проверка размаха рук
        if (self.wingspan >= 200) and (self.wingspan <= 250):
            count_goods += 1
            if self.wingspan > 225:
                top_half += 1
        elif self.wingspan > 250:
            count_sgoods += 1
            green_unicorn = True
        # Проверка среднего кол-ва очков
        if (self.avg_points >= 10) and (self.avg_points <= 20):
            count_goods += 1
            if self.avg_points > 15:
                top_half += 1
        elif self.avg_points > 20:
            count_sgoods += 1
        # Проверка среднего кол-ва подборов
        if (self.avg_takes >= 2) and (self.avg_takes <= 6):
            count_goods += 1
            if self.avg_takes > 4:
                top_half += 1
        elif self.avg_takes > 6:
            count_sgoods += 1
        # Проверка среднего кол-ва передач
        if (self.avg_passes >= 3) and (self.avg_passes <= 7):
            count_goods += 1
            if self.avg_passes > 4:
                top_half += 1
        elif self.avg_passes > 7:
            count_sgoods += 1
```

```

# Выбор категории
# Проверка на категорию 0
if green_unicorn:
    if count_sgoods > 2:
        return 0
# Проверка на категорию 1
if ((count_sgoods > 1) and (top_half > 0)) or (((count_goods +
count_sgoods) > 4) and ((count_sgoods + top_half) > 2)):
    return 1
# Проверка на категорию 2
if ((count_sgoods > 0) and (top_half > 0)) or (top_half > 2):
    return 2

return 3

```

Далее представлен код файла main.py, где реализуется сам драфт на основе входного файла in.txt.

### main.py

```

import player

if __name__ == '__main__':
    file = open('in.txt', 'r').readlines()
    out_string = ''
    a = 0
    b = 0
    c = 0
    d = 0
    e = 0

    for i in range(len(file)):
        if i == 0:
            count = int(file[0])
        else:
            if a == 0:
                a = int(file[i])
            elif b == 0:
                b = int(file[i])
            elif c == 0:
                c = int(file[i])
            elif d == 0:
                d = int(file[i])
            elif e == 0:
                e = int(file[i])
            else:
                pl = player.player(a, b, c, d, e)
                out_string += str(pl.get_player_category()) + '\n'
                a = int(file[i])
                b = 0
                c = 0
                d = 0
                e = 0

    pl = player.player(a, b, c, d, e)
    out_string += str(pl.get_player_category()) + '\n'

    open('out.txt', 'w').write(out_string)

```

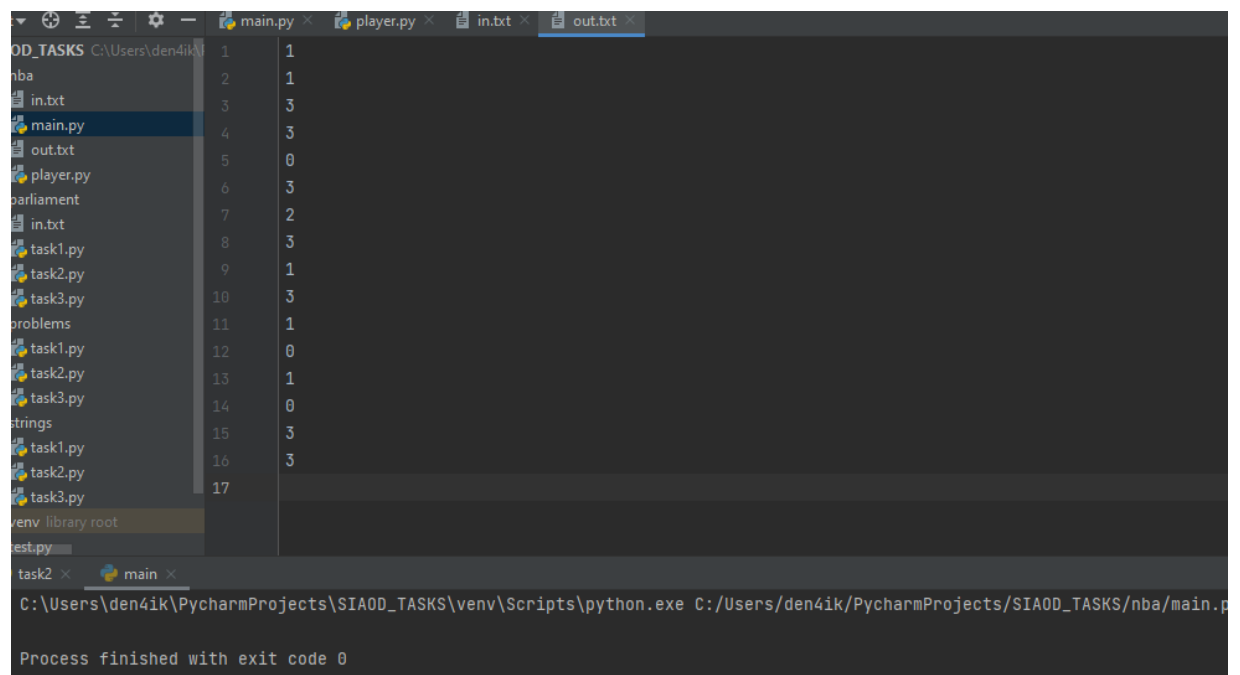
in.txt

16  
198  
231  
21  
4  
6  
211  
241  
11  
3  
8  
180  
205  
14  
5  
4  
170  
198  
6  
4  
2  
233  
252  
25  
8  
4  
212  
219  
12  
4  
5  
191  
209  
11  
8  
5  
187  
199  
15  
6  
4  
220  
237  
19  
7  
5  
190  
214  
5  
3  
2  
201  
235  
15  
5  
8  
241  
260  
28  
8  
2  
199



```
221
21
5
8
219
255
23
7
5
200
220
18
5
4
195
227
11
6
3
```

Ниже представлен результат работы программы.



The screenshot shows the PyCharm IDE interface. On the left, the 'Project' tool window displays a file tree for a project named 'SIA00\_TASKS'. The tree includes a folder 'nba' and several files: 'in.txt', 'main.py', 'out.txt', 'player.py', 'parliament', 'in.txt', 'task1.py', 'task2.py', 'task3.py', 'problems', 'task1.py', 'task2.py', 'task3.py', 'strings', 'task1.py', 'task2.py', 'task3.py', 'venv library root', and 'test.py'. The 'main.py' file is selected. The main editor window shows the 'out.txt' file, which contains a list of numbers: 1, 1, 3, 3, 0, 3, 2, 3, 1, 3, 1, 0, 1, 0, 3, 3. The bottom console window shows the command to run the program: 'C:\Users\den4ik\PycharmProjects\SIA00\_TASKS\venv\Scripts\python.exe C:/Users/den4ik/PycharmProjects/SIA00\_TASKS/nba/main.py'. The console output shows 'Process finished with exit code 0'.

Рис. 1 – Результат работы программы.

out.txt

```
1
1
3
3
0
3
2
3
1
3
1
0
1
0
3
3
```

## 2.parliament

Далее рассмотрим следующие 3 задачи.

Задача 1. Вам дана запись о событиях на сессии парламента. Каждое событие является либо внесением нового законопроекта, либо голосованием за какой-то законопроект, причём в обоих случаях известен номер партии, этот законопроект предложившей. События даны в том порядке, в котором они происходили. Проверьте, может ли данная запись о событиях соответствовать какому-либо заседанию, удовлетворяющему порядку проведения, описанному выше.

Ниже представлен код решения задачи task1.py а также входной файл in.txt.

task1.py

```
def check(file):
    f = open(file, 'r').readlines()
    count = int(f[0])
    stack = []

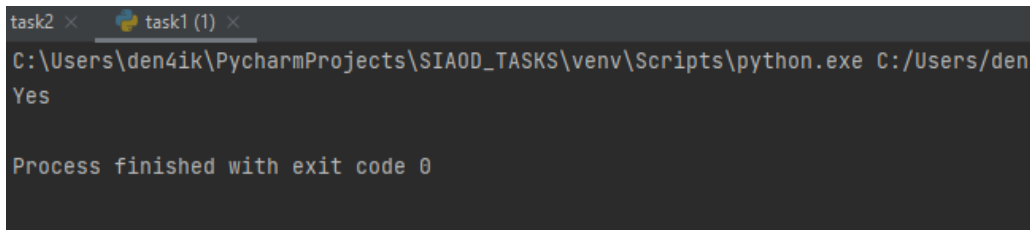
    for i in range(1, count + 1):
        if f[i][0] == 'A': # Добавляем проект
            stack.append(str(f[i][3]))
        else: # Закрываем проект
            if len(stack) > 0:
                if stack[len(stack) - 1] == str(f[i][4]):
                    stack.pop()
            else:
                return 'No'
    if len(stack) > 0:
        return 'No'
    else:
        return 'Yes'

if __name__ == '__main__':
    print(check('in.txt'))
```

in.txt

32  
Addb  
Addc  
Addt  
Addf  
Addp  
Addo  
Adda  
Addx  
Addx  
Votex  
Votex  
Addb  
Addx  
Votex  
Voteb  
Votea  
Voteo  
Votep  
Addz  
Addp  
Addr  
Voter  
Addl  
Votel  
Votep  
Votez  
Votef  
Votet  
Addb  
Voteb  
Votec  
Voteb

На рисунке 2 представлен результат в выполнения программы.



```
task2 × task1 (1) ×
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/den4ik/task2.py
Yes
Process finished with exit code 0
```

Рис. 2 – Результат выполнения программы.

Задача 2. Задан правильный N-угольник. Требуется выбрать наименьшее количество его вершин, которые также образуют правильный многоугольник.

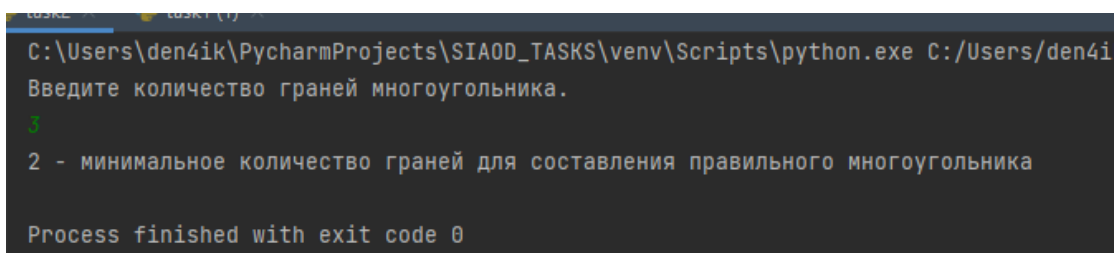
Ниже представлен код файла task2.py который выполняет данную задачу.

task2.py

```
def count_versh(N):
    if (N < 3) or (N > 10**12):
        return 'Число не входит в заданный диапазон.'
    else:
        if N%2 == 0:
            return str(N // 2)
        else:
            return str((N+1) // 2)

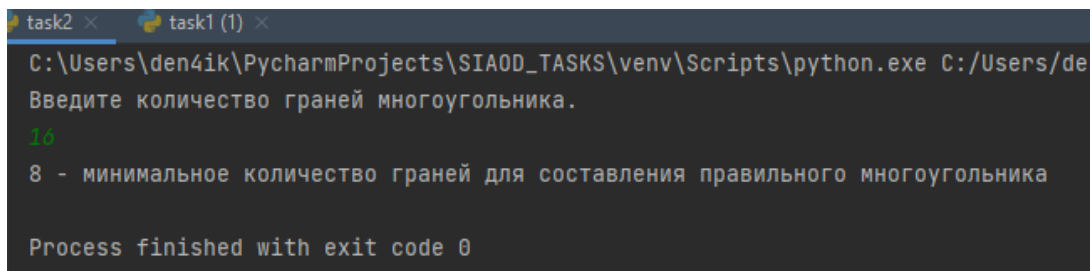
if name == ' main ':
    N = int(input('Введите количество граней многоугольника.\n'))
    print(count_versh(N) + ' - минимальное количество граней для составления правильного многоугольника')
```

На рисунках 3, 4 представлены результаты работы программы.



```
task2 × task1 (1) ×
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/den4ik/task2.py
Введите количество граней многоугольника.
3
2 - минимальное количество граней для составления правильного многоугольника
Process finished with exit code 0
```

Рис. 3 – Результат работы программы.



```
task2 x task1 (1) x
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/de
Введите количество граней многоугольника.
16
8 - минимальное количество граней для составления правильного многоугольника

Process finished with exit code 0
```

Рис. 4 – Результат работы программы.

Задача 3. На форуме, на котором обсуждаются задачи олимпиад по информатике, ввели следующий аналог капчи. Участнику выдаётся строка из  $N$  десятичных цифр (без ведущих нулей). В качестве ответа требуется ввести такое основание системы счисления  $B$ , что в этой системе счисления выданная запись будет соответствовать составному числу (назовем его  $D$ ), а также число  $X$ , большее 1 и меньшее  $D$ , являющееся делителем  $D$ .

При этом  $B$  и  $X$  не должны превосходить 109.

По заданной строке десятичных цифр найдите любую пару чисел  $B$  и  $X$ , удовлетворяющую ограничениям, или ответьте, что решения в заданных ограничениях не существует.

Ниже представлен код файла `task3.py` выполняющий приведенную выше задачу.

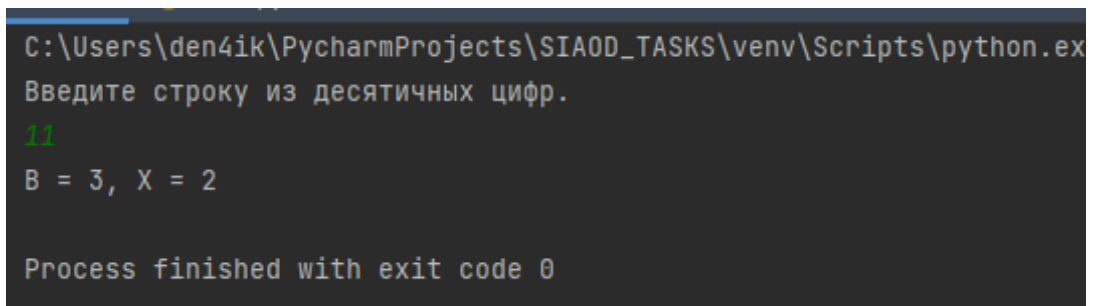
`task3.py`

```
def thc(s1):
    chislo10 = int(s1)
    for base in range(2, 10**9):
        if not (str(chislo10).__contains__(str(base))):
            for i in range(chislo10):
                num = in_any_base(i, base)
                if str(num) == str(chislo10):
                    D = i
                    B = -1
                    if (base >= 2) and (base <= 10 ** 9):
                        B = base
                    x = D//2
                    if (x >= 2) and (x <= 10 ** 9):
                        X = x
                    else:
                        X = -1
                    if (not (B == -1)) and (not (X == -1)):
                        return 'B = ' + str(B) + ', X = ' + str(X)
    return -1
```

```
def in_any_base(num, base):
    newNum = ''
    while num > 0:
        newNum = str(num % base) + newNum
        num //= base
    return newNum

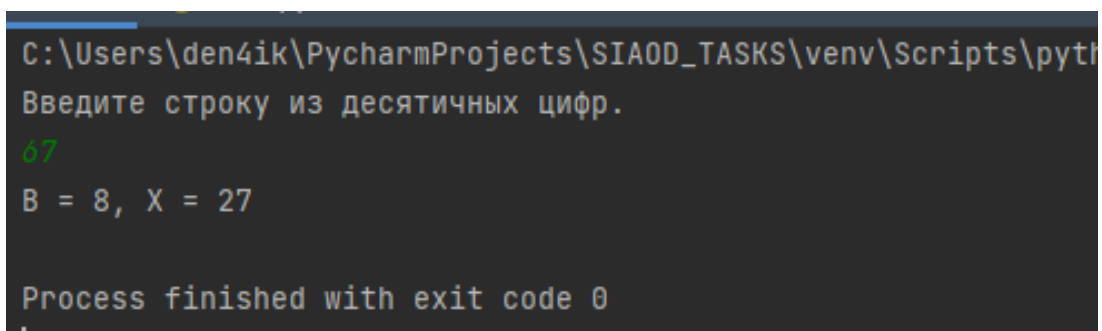
if __name__ == '__main__':
    s1 = input('Введите строку из десятичных цифр.\n')
    print(thc(s1))
```

На рисунках 5, 6 представлены результаты выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe
Введите строку из десятичных цифр.
11
B = 3, X = 2
Process finished with exit code 0
```

Рис. 5 – Результат выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe
Введите строку из десятичных цифр.
67
B = 8, X = 27
Process finished with exit code 0
```

Рис. 6 – Результат выполнения программы.

### 3.strings

Далее рассмотрим следующие 3 задачи.

Задача 1. Даны две строки:  $s1$  и  $s2$  с одинаковым размером, проверьте, может ли некоторая перестановка строки  $s1$  “победить” некоторую перестановку строки  $s2$  или наоборот. Строка  $x$  может “победить” строку  $y$  (обе имеют размер  $n$ ), если  $x[i] \geq y[i]$  (в алфавитном порядке) для всех  $i$  от 0 до  $n-1$ .

Ниже представлен код файла `task1.py` выполняющий решение приведенной выше задачи.

`task1.py`

```
import itertools

def can_win(s1, s2):
    if not (len(s1) == len(s2)):
        return False
    if (len(s1) > 10**5) or (len(s1) < 1) or (len(s2) > 10**5) or (len(s2) < 1):
        return False
    else:
        n = len(s1)
        perm1 = itertools.permutations(s1)
        perm2 = itertools.permutations(s2)
        for a in perm1:
            for b in perm2:
                s1_syms = []
                s2_syms = []
                s1_wins = 0
                s2_wins = 0
                for z in a:
                    s1_syms.append(z)
                for z in b:
                    s2_syms.append(z)

                for i in range(0, len(s1_syms)):
                    c = s1_syms[i]
                    d = s2_syms[i]
                    if ord(c) >= ord(d):
                        s1_wins += 1
                        if s1_wins == n:
                            return True
                    else:
                        s1_wins = 0
                    if ord(c) <= ord(d):
                        s2_wins += 1
                        if s2_wins == n:
                            return True
                    else:
                        s2_wins = 0

        return False
```

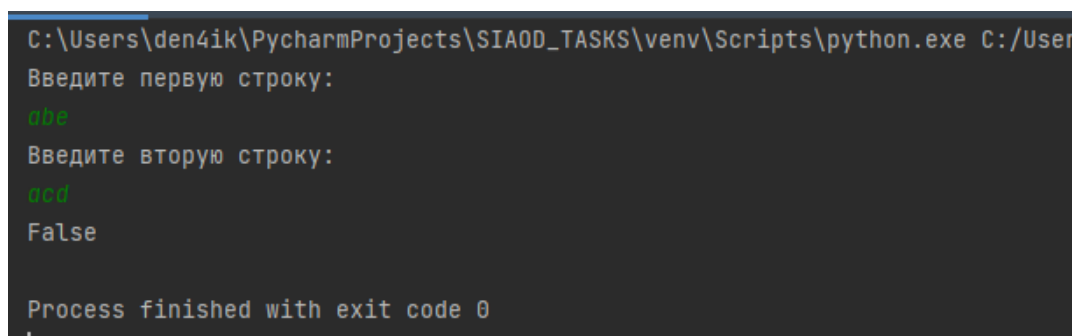
```
if __name__ == '__main__':  
    print('Введите первую строку:')  
    s1 = input()  
    print('Введите вторую строку:')  
    s2 = input()  
    print(can_win(s1, s2))
```

Ниже на рисунках 7, 8 представлены результаты выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/  
Введите первую строку:  
abc  
Введите вторую строку:  
xua  
True  
  
Process finished with exit code 0
```

Рис. 7 – Результат выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/User  
Введите первую строку:  
abe  
Введите вторую строку:  
acd  
False  
  
Process finished with exit code 0
```

Рис. 8 – Результат выполнения программы.



Задача 2. Дана строка s, вернуть самую длинную полиндромную подстроку в s.

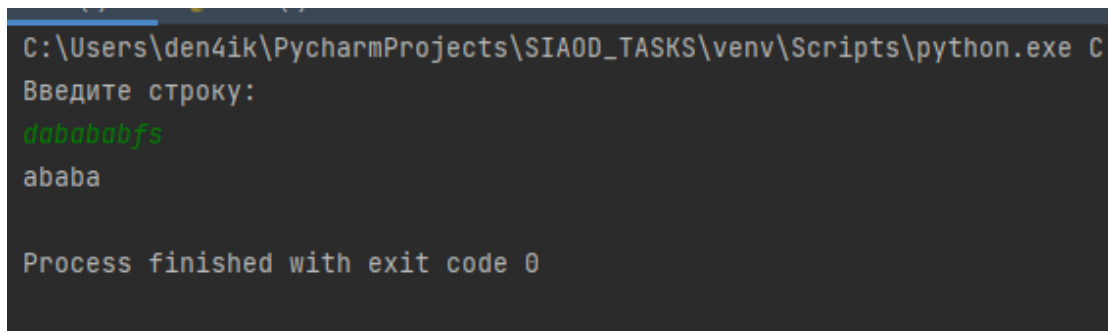
Ниже в представлен код файла task2.py выполняющий решение данной задачи.

task2.py

```
def find_pol(string):
    pol_list = []
    res = [string[i: j] for i in range(len(string))
           for j in range(i + 1, len(string) + 1)]
    max = 0
    for i in res:
        if i == i[::-1]:
            if len(i) > max:
                max = len(i)
                if len(pol_list) > 0:
                    pol_list.pop()
                pol_list.append(i)
    if len(pol_list) > 0:
        return pol_list.pop()
    else:
        return 'No polindroms'

if __name__ == '__main__':
    print('Введите строку:')
    s1 = str(input())
    print(find_pol(s1))
```

На рисунке ниже представлен результат выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C
Введите строку:
dabababfs
ababa

Process finished with exit code 0
```

Рис. 9 – Результат выполнения программы.

Задача 3. Вернуть количество отдельных непустых подстрок текста, которые могут быть записаны как конкатенация некоторой строки с самой собой (т.е. она может быть записана, как  $a + a$ , где  $a$  - некоторая строка).

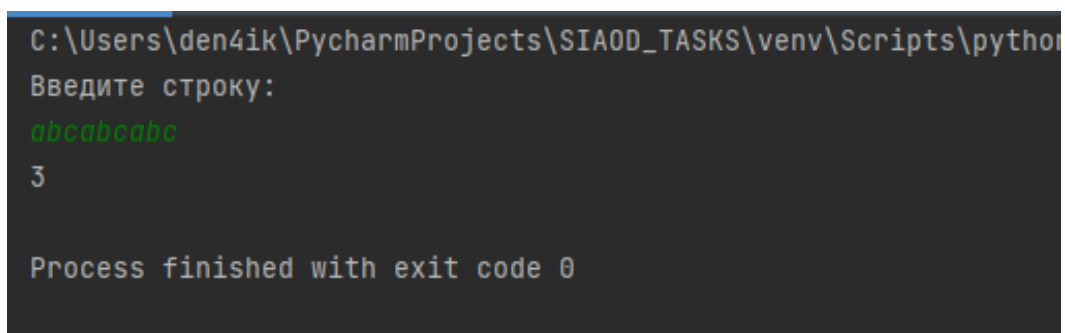
Ниже представлен код файла task3.py выполняющий решение приведенной выше задачи.

task3.py

```
def count_of_conc(string):
    counter = 0
    res = [string[i: j] for i in range(len(string))
           for j in range(i + 1, len(string) + 1)]
    lfl = []
    for i in res:
        if (len(i) % 2) == 0:
            s1 = i[0:(len(i)//2)]
            s2 = i[(len(i)//2):len(i)]
            if s1 == s2:
                if not (lfl.__contains__(s1+s2)):
                    counter += 1
                    lfl.append(s1+s2)
    return counter

if __name__ == '__main__':
    print('Введите строку:')
    s1 = input()
    print(count_of_conc(s1))
```

На рисунке 10 представлен результат выполнения программы.



```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python
Введите строку:
abcabcabc
3
Process finished with exit code 0
```

Рис. 10 – Результат выполнения программы.

#### 4.problems

Далее рассмотрим следующие 3 задачи.

Задача 1. Массив A состоит из целых положительных чисел длин отрезков. Составьте из трех отрезков такой треугольник, чтобы его периметр был максимально возможным. Если невозможно составить треугольник с положительной площадью функция возвращает 0.

Ниже представлен код файла task1.py выполняющий решение данной задачи.

task1.py

```
def max_p(list):
    if (len(list) < 3) or (len(list) > 10000):
        return 0
    max_list = []
    temp_max = list[0]
    while len(list) > 0:
        while len(max_list) < 3:
            for i in list:
                if (i < 1) or (i > 10**6):
                    return 0
                if i > temp_max:
                    temp_max = i
            if list.__contains__(temp_max):
                list.remove(temp_max)
                max_list.append(temp_max)
                temp_max = 0
            else:
                temp = list.pop()
                max_list.append(temp)
        max = max_element(max_list)
        max_list.remove(max)
        if (max_list[0] + max_list[1]) > max:
            max_p = max + max_list[0] + max_list[1]
            return max_p
    return 0

def max_element(list):
    max = 1
    for i in list:
        if i > max:
            max = i
    return max

def min_element(list):
    min = 1
    for i in list:
        if i < min:
            min = i
    return min
```

```

if __name__ == '__main__':
    arr = []
    print('Введите массив.')
    inp = input()
    while not (inp == 'stop'):
        try:
            inp = int(inp)
            if inp == int(inp):
                arr.append(inp)
        except:
            print('Введеное значение не является числом')
            inp = input()
    print(max_p(arr))

```

На рисунках 11, 12 представлены результаты выполнения программы.

```

C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/den4ik/Pychar
Введите массив.
4
6
13
2
1
8
5
stop
30
Process finished with exit code 0

```

Рис. 11 – Результат выполнения программы.

```

C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe C:/Users/den4ik/Pychar
Введите массив.
1
2
1
stop
0
Process finished with exit code 0

```

Рис. 12 – Результат выполнения программы.

Задача 2. Дан массив неотрицательных целых чисел `nums`. Расположите их в таком порядке, чтобы вместе они образовали максимально возможное число.

Ниже представлен код файла `task2.py` выполняющий решение приведенной выше задачи.

`task2.py`

```
def max_chislo(arr):
    stck1 = []
    stck2 = []
    # Проходимся по исходному массиву и сортируем ценные числа в стек
    for i in range(0, len(arr)):
        worth = 0
        s1 = str(arr[i])
        for a in s1:
            worth += int(a)
        worth = worth/len(s1)
        if len(stck1) <= 0: # Если stck1 пуст просто записываем в него
            рассматриваемое число
            stck1.append([s1, worth])

        # Загонем в stck1 числа с ценностью от наименьшей к наибольшей
        else:
            last = stck1.pop()
            if worth > last[1]: # При этом условии записываем
                рассматриваемое число в stck1
                stck1.append(last)
                stck1.append([s1, worth])

            elif worth < last[1]: # При этом условии перегоняем элементы из
                stck1 в stck2 пока не найдем подходящее место
                while worth < last[1]:
                    stck2.append(last)
                try:
                    last = stck1.pop()
                    if worth > last[1]:
                        stck1.append(last)
                        stck1.append([s1, worth])
                    while len(stck2) > 0:
                        stck1.append(stck2.pop())

                elif worth == last[1]: # При этом условии сравниваем
                    количество цифр в числах, большую ценность имеет короткое
                    if len(s1) < len(last[0]): # Тут у
                        рассматриваемого числа ценность больше
                            stck1.append(last)
                            stck1.append([s1, worth])
                    elif len(s1) > len(last[0]):
                        stck1.append([s1, worth])
                        stck1.append(last)
                    else:
                        if int(s1) > int(last[0]):
                            stck1.append(last)
                            stck1.append([s1, worth])
                        else:
                            stck1.append([s1, worth])
                            stck1.append(last)
                    while len(stck2) > 0:
```

```

        stck1.append(stck2.pop())

    except:
        stck1.append([s1, worth])
        while len(stck2) > 0:
            stck1.append(stck2.pop())
        break

    elif worth == last[1]: # При этом условии сравниваем количество
        цифр в числах, большую ценность имеет короткое
        if len(s1) < len(last[0]): # Тут у рассматриваемого числа
            ценность больше
            stck1.append(last)
            stck1.append([s1, worth])
        elif len(s1) > len(last[0]):
            stck1.append([s1, worth])
            stck1.append(last)
        else:
            if int(s1) > int(last[0]):
                stck1.append(last)
                stck1.append([s1, worth])
            else:
                stck1.append([s1, worth])
                stck1.append(last)

big_chislo = ''
while len(stck1) > 0:
    big_chislo += str(stck1.pop()[0])

return big_chislo

if __name__ == '__main__':
    arr = []
    inp = input()
    while not (inp == 'stop'):
        try:
            inp = int(inp)
            if (inp > 10 ** 9) or (inp < 0):
                print('Введеное значение должно быть от 0 до 10^9.')
            else:
                if inp == int(inp):
                    arr.append(inp)
        except:
            print('Введеное значение не является числом')
            inp = input()

    if not (len(arr) < 1) or (len(arr) > 100):
        print(max_chislo(arr))
    else:
        print('Длина массива неверна')

```

На следующих трех рисунках представлены результаты выполнения программы.

```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe 0
1
10
stop
110

Process finished with exit code 0
```

Рис. 13 – Результат выполнения программы.

```
C:\Users\den4ik\PycharmProjects\SIAOD_TASKS\venv\Scripts\python.exe
10
1
stop
110

Process finished with exit code 0
```

Рис. 14 – Результат выполнения программы.

```
2
16
9
52
13
stop
9521624213

Process finished with exit code 0
```

Рис. 15 – Результат выполнения программы.

Задача 3. Дана матрица `mat` размером  $m * n$ , значения целочисленные. Напишите функцию, сортирующую каждую диагональ матрицы по возрастанию и возвращающую получившуюся матрицу

Ниже представлен код файла `task3.py` выполняющий решения приведенной задачи.

`task3.py`

```
def sort_d(A):
    try:
        m = len(A)
        n = len(A[0])
        if (m > 100) or (m < 1) or (n > 100) or (n < 1):
            return print('Неверный размер матрицы. (1 <= m, n <= 100)')

        temp_list = []
        closed_list = []

        for i in range(0, m):
            for j in range(0, n):
                if (A[i][j] < 1) or (A[i][j] > 100):
                    return print('В матрице содержится неподходящее число. (1 <= mat[i][j] <= 100)')
                if not (closed_list.__contains__([i, j])): #Проверяем только те ячейки которых нет в списке закрытых (просмотренных)
                    temp_list.append(A[i][j])
                    new_i = i + 1
                    new_j = j + 1
                    closed_list.append([i, j])
                    while (new_i < m) and (new_j < n) and (not (closed_list.__contains__([new_i, new_j]))): #Пока по диагонали есть ячейки записываем их в временный список
                        closed_list.append([new_i, new_j])
                        temp_list.append(A[new_i][new_j])
                        new_i += 1
                        new_j += 1
                    temp_list.sort()
                    new_i = i
                    new_j = j

            for k in range(0, len(temp_list)): # Изменяем диагональ в матрице только что отсортированной диагональю
                A[new_i][new_j] = temp_list[k]
                new_i += 1
                new_j += 1
            temp_list = []

        print(A)
    except:
        print('Что то пошло не так...')

if __name__ == '__main__':
    mat = [[11, 25, 66, 1, 69, 7], [23, 55, 17, 45, 15, 52], [75, 31, 36, 44, 58, 8], [22, 27, 33, 25, 68, 4], [84, 28, 14, 11, 5, 50]]
    sort_d(mat)
```



На рисунках 16, 17 представлены результаты выполнения программы.

```
C:\Users\den4ik\PycharmProjects\SIA0D_TASKS\venv\Scripts\python.exe C:/Users/den4ik/PycharmProjects/SIA0D_TASKS/problems/task3.py
Исходная матрица:
[[11, 25, 66, 1, 69, 7], [23, 55, 17, 45, 15, 52], [75, 31, 36, 44, 58, 8], [22, 27, 33, 25, 68, 4], [84, 28, 14, 11, 5, 50]]

Отсортированная матрица.
[[5, 17, 4, 1, 52, 7], [11, 11, 25, 45, 8, 69], [14, 23, 25, 44, 58, 15], [22, 27, 31, 36, 50, 66], [84, 28, 75, 33, 55, 68]]

Process finished with exit code 0
```

Рис. 16 – Результат выполнения программы.

```
C:\Users\den4ik\PycharmProjects\SIA0D_TASKS\venv\Scripts\python.exe C:/Users/den4ik/PycharmProjects/SIA0D_TASKS/problems/task3.py
Исходная матрица:
[[62, 95, 2, 58, 9, 79], [11, 21, 10, 91, 13, 14], [86, 27, 7, 42, 5, 4], [19, 27, 69, 21, 68, 41], [92, 25, 98, 24, 51, 2]]

Отсортированная матрица.
[[7, 2, 2, 4, 9, 79], [11, 21, 10, 5, 13, 14], [27, 24, 21, 42, 41, 58], [19, 86, 27, 51, 68, 91], [92, 25, 98, 69, 62, 95]]

Process finished with exit code 0
```

Рис. 17 – Результат выполнения программы.

### 3. ВЫВОД

В ходе данной курсовой работы были отточены навыки, приобретенные в ходе курса Структур и Алгоритмов Обработки Данных, на примере приведенных заданий.