

Федеральное агентство связи
Ордена Трудового Красного Знамени федеральное государственное
бюджетное образовательное учреждение высшего образования
«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №4
по дисциплине «Структура и алгоритмы обработки данных»
«Реализация стека/дека»

Выполнил студент
группы БФИ1902
Кочеринский Н.В.
Проверил: Мкртчян Г.М.

Москва 2021

Оглавление

1 Задание на лабораторную работу.....	3
2 Решение лабораторной работы.....	4
2.1 Содержание текстовых файлов.	4
2.2 Листинг и результаты лабораторной работы.	6

1 Задание на лабораторную работу.

А) Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.

Б) Дек содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь деком, расшифровать текст. Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в деке по часовой стрелке через один.

В) Даны три стержня и n дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести n дисков со стержня А на стержень С, сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила: - на каждом шаге со стержня на стержень переносить только один диск; - диск нельзя помещать на диск меньшего размера; - для промежуточного хранения можно использовать стержень В. Реализовать алгоритм, используя три стека вместо стержней А, В, С. Информация о дисках хранится в исходном файле.

Г) Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя стек.

Д) Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя дек.

Е) Дан файл из символов. Используя стек, за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.

Ж) Дан файл из целых чисел. Используя дек, за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.

З) Дан текстовый файл. Используя стек, сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.

И) Дан текстовый файл. Используя стек, вычислить значение логического выражения, записанного в текстовом файле в следующей форме: $\langle \text{ЛВ} \rangle ::= T \mid F \mid (N\langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle A \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle X \langle \text{ЛВ} \rangle) \mid (\langle \text{ЛВ} \rangle O \langle \text{ЛВ} \rangle)$, где буквами обозначены логические константы и операции: T – True, F – False, N – Not, A – And, X – Xor, O – Or.

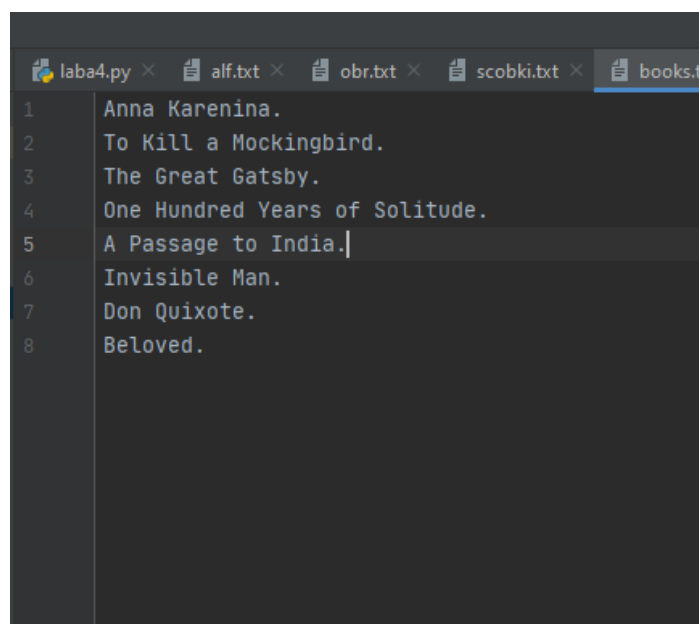
К) Дан текстовый файл. В текстовом файле записана формула следующего вида: $\langle \text{Формула} \rangle ::= \langle \text{Цифра} \rangle \mid M(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle) \mid N(\langle \text{Формула} \rangle, \langle \text{Формула} \rangle)$ $\langle \text{Цифра} \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$ где буквами обозначены функции: M – определение максимума, N – определение минимума. Используя стек, вычислить значение заданного выражения.

Л) Дан текстовый файл. Используя стек, проверить, является ли содержимое текстового файла правильной записью формулы вида: $\langle \text{Формула} \rangle ::= \langle \text{Терм} \rangle \mid \langle \text{Терм} \rangle + \langle \text{Формула} \rangle \mid \langle \text{Терм} \rangle - \langle \text{Формула} \rangle \mid \langle \text{Терм} \rangle \mid \langle \text{Имя} \rangle \mid (\langle \text{Формула} \rangle) \mid \langle \text{Имя} \rangle ::= x \mid y \mid z$

2 Решение лабораторной работы

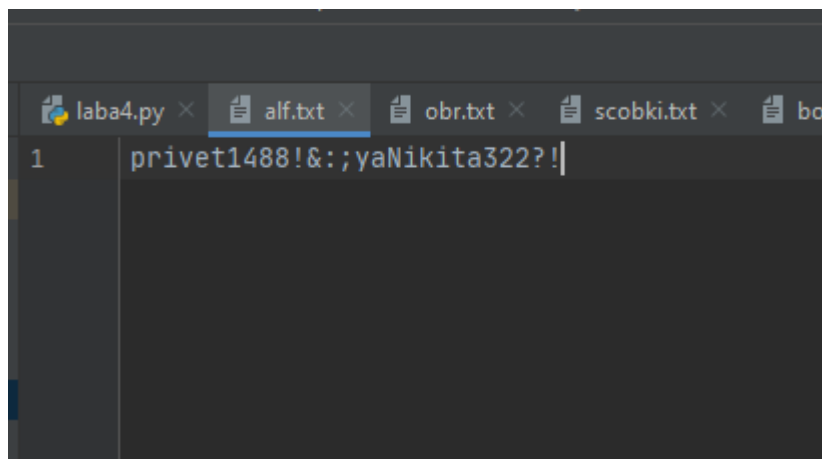
2.1 Содержание текстовых файлов.

На рисунках 1-4 представлено содержимое текстовых файлов, которые необходимы для выполнения лабораторной работы.



```
laba4.py x  alf.txt x  obr.txt x  scobki.txt x  books.t
1  Anna Karenina.
2  To Kill a Mockingbird.
3  The Great Gatsby.
4  One Hundred Years of Solitude.
5  A Passage to India.
6  Invisible Man.
7  Don Quixote.
8  Beloved.
```

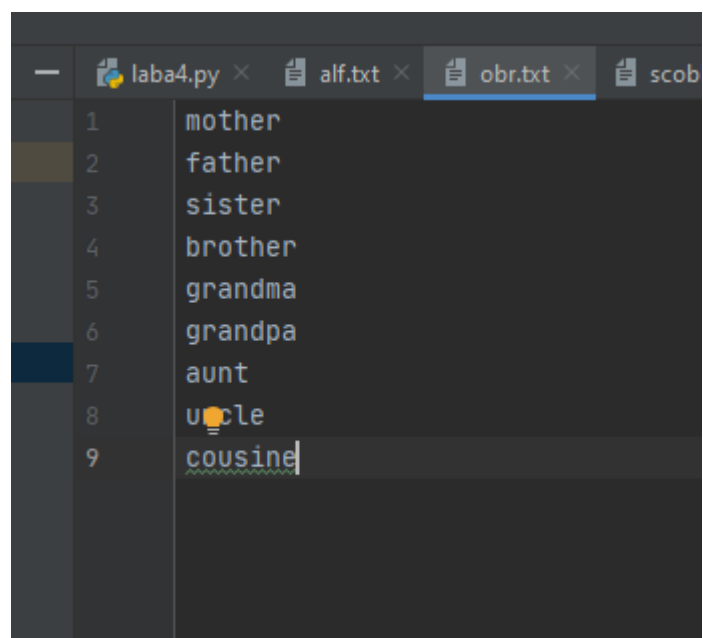
Рисунок 1 – Текст файла books.txt



The screenshot shows a code editor with several tabs: 'laba4.py', 'alf.txt', 'obr.txt', 'scobki.txt', and 'books.txt'. The 'books.txt' tab is active, and the text 'privet1488!&;yaNikita322?!' is visible on line 1.

```
1  privet1488!&;yaNikita322?!|
```

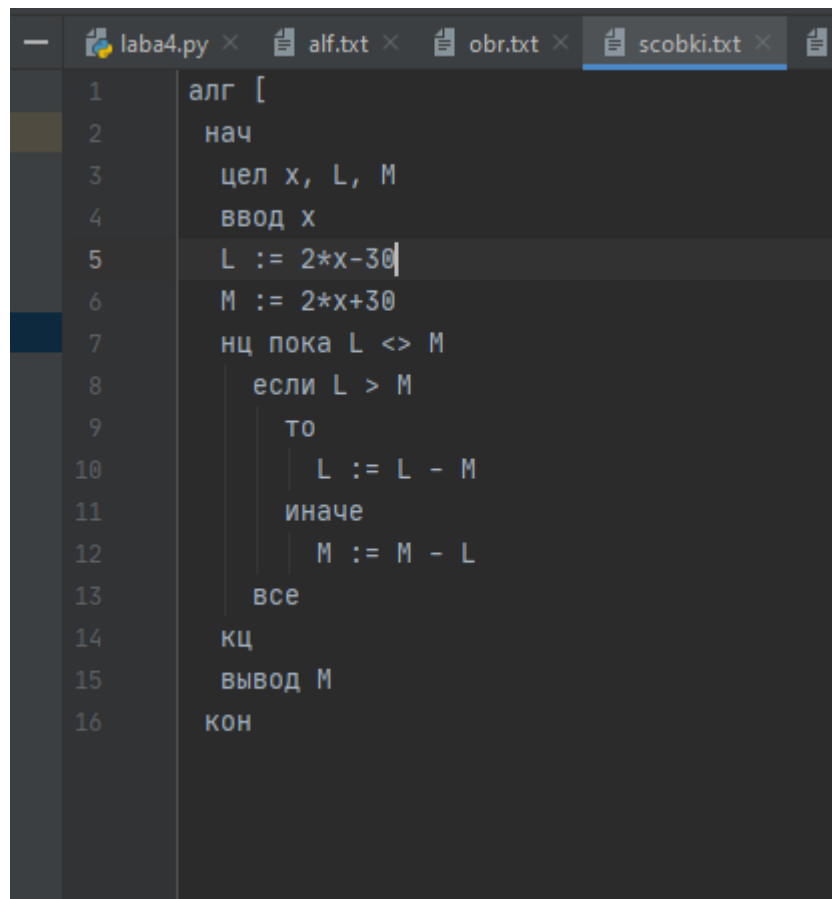
Рисунок 2 – Текст файла alf.txt



The screenshot shows a code editor with tabs: 'laba4.py', 'alf.txt', 'obr.txt', and 'scobki.txt'. The 'obr.txt' tab is active, and the text 'mother', 'father', 'sister', 'brother', 'grandma', 'grandpa', 'aunt', 'uncle', and 'cousine' is visible on lines 1 through 9.

```
1  mother
2  father
3  sister
4  brother
5  grandma
6  grandpa
7  aunt
8  uncle
9  cousin|
```

Рисунок 3 – Текст файла obr.txt



```
1  алг [  
2  нач  
3  цел x, L, M  
4  ввод x  
5  L := 2*x-30  
6  M := 2*x+30  
7  нц пока L <> M  
8  |если L > M  
9  |то  
10 |  L := L - M  
11 |иначе  
12 |  M := M - L  
13 |все  
14 кц  
15 вывод M  
16 кон
```

Рисунок 4 – Текст файла scobki.txt

2.2 Листинг и результаты лабораторной работы.

Для выполнения лабораторной работы необходимо выполнить задания, которые были показаны в пункте 1, подпунктах А-Л. На рисунках 5-9 представлен результат работы программы.

Листинг 1

```
# -*- coding: utf-8 -*-  
import random
```

```
class ListNode:
```

```
    # Связный список с ссылками на предыдущий и следующий элемент
```

```
    def __init__(self, value=None):
```

```
        self.value = value
```

```
        self.right = None
```

```
        self.left = None
```

```
class Stack:
```

```
    def __init__(self):
```

```

self.head = LinkedNode()
self.size = 0

# Проверка на пустату
def is_empty(self):
    return self.size == 0

# Добавление
def push(self, value):
    if self.size > 0:
        node = LinkedNode(value)
        node.right = self.head
        self.head = node
    else:
        self.head.value = value
    self.size += 1

# Удаление
def pop(self):
    if self.is_empty():
        raise Exception("Popping from an empty stack")
    remove = self.head
    if self.size > 1:
        self.head = remove.right
    self.size -= 1
    return remove.value

def peek(self):
    if self.is_empty():
        raise Exception("Popping from an empty stack")
    return self.head.value

def __len__(self):
    return self.size

def reverse(self):
    current = self.head
    prev = None
    next = None

    while current is not None:
        next = current.right
        current.right = prev
        prev = current
        current = next

```

```
self.head = prev
```

```
class Deque:
```

```
    def __init__(self):  
        self.head = LinkedNode()  
        self.tail = self.head  
        self.size = 0
```

```
    # Проверка на пустоту
```

```
    def is_empty(self):  
        return self.size == 0
```

```
    # Добавление в конец
```

```
    def push_left(self, value):  
        if self.size > 0:  
            node = LinkedNode(value)  
            node.right = self.tail  
            self.tail.left = node  
            self.tail = node  
        else:  
            self.tail.value = value  
        self.size += 1
```

```
    # Добавление в начало
```

```
    def push(self, value):  
        if self.size > 0:  
            node = LinkedNode(value)  
            node.left = self.head  
            self.head.right = node  
            self.head = node  
        else:  
            self.head.value = value  
        self.size += 1
```

```
    # Удаление в конец
```

```
    def pop_left(self):  
        if self.is_empty():  
            raise Exception("Popping from an empty deque")  
        remove = self.tail  
        if self.size > 1:  
            self.tail = remove.right  
        self.size -= 1  
        return remove.value
```



```

# Удаление в начале
def pop(self):
    if self.is_empty():
        raise Exception("Popping from an empty deque")
    remove = self.head
    if self.size > 1:
        self.head = remove.left
    self.size -= 1
    return remove.value

def peek(self):
    if self.is_empty():
        raise Exception("Popping from an empty deque")
    return self.head.value

def peek_left(self):
    if self.is_empty():
        raise Exception("Popping from an empty deque")
    return self.tail.value

def __len__(self):
    return self.size

```

#Задание 1

```

print("\nЗадание №1\n")
with open('books.txt', 'r') as books:
    d1 = Deque()
    d2 = Deque()
    for book in books:
        d1.push(book)
    while not d1.is_empty():
        a = d1.pop()
        while not d2.is_empty() and d2.peek() > a:
            d1.push_left(d2.pop())
        d2.push(a)
    while not d2.is_empty():
        print(d2.pop_left())

```

#Задание 2

```

print("\nЗадание №2\n")

alphabet = list('абвгдеёжзийклмнопрстуфхцчшщъыьэюя')
random.shuffle(alphabet)
alphabet = "".join(alphabet)

```

```
print(alphabet)
keyRing = Deque()
for letter in alphabet:
    keyRing.push(letter)
```

```
def encode_char(c):
    for i in range(len(keyRing)):
        x = keyRing.pop_left()
        if x == c:
            keyRing.push(x)
            val = keyRing.pop_left()
            keyRing.push(val)
            return val
    keyRing.push(x)
```

```
def decode_char(c):
    for i in range(len(keyRing)):
        x = keyRing.pop()
        if x == c:
            keyRing.push_left(x)
            val = keyRing.pop()
            keyRing.push_left(val)
            return val
    keyRing.push_left(x)
```

```
text = 'Я помню чудное мгновенье: передо мной явилась ты, как мимолетное
виденье, как гений чистой красоты. В томленьях грусти безнадежной, в
тревогах шумной суеты, звучал мне долго голос нежный и снились милые
черты.'.lower()
```

```
encoded = ""
for letter in text:
    if encoded_letter := encode_char(letter):
        encoded += encoded_letter
    else:
        encoded += letter
```

```
print(encoded)
```

```
decoded = ""
for letter in encoded:
    if decoded_letter := decode_char(letter):
        decoded += decoded_letter
```

```

    else:
        decoded += letter
print(decoded)
#Задание 3
print("\nЗадание №3\n")
A = Stack()
B = Stack()
C = Stack()

disks = 10

for i in range(disks, 0, -1):
    A.push(i)

def move(a, b):
    if len(a) == 0 and len(b) > 0:
        a.push(b.pop())
    elif len(a) > 0 and len(b) == 0:
        b.push(a.pop())
    elif a.peek() > b.peek():
        a.push(b.pop())
    else:
        b.push(a.pop())

if disks % 2 == 0:
    while len(C) != disks:
        move(A, B)
        move(A, C)
        move(B, C)
else:
    while len(C) != disks:
        move(A, C)
        move(A, B)
        move(B, C)

while not C.is_empty():
    print(C.pop())
#Задание 4
print("\nЗадание №4\n")
def check_brackets(string):
    bracket_stack = Stack()
    for i in string:
        if i == '(':
            bracket_stack.push(i)
        elif i == ')':

```

```

        if bracket_stack.is_empty():
            return False
        bracket_stack.pop()
    return bracket_stack.is_empty()

print(check_brackets('()()((()()()()()'))
print(check_brackets('((()()()()()()()()()()'))
#Задание 5
print("\nЗадание №5\n")
check = Deque()

```

```

with open('scobki.txt', 'r') as brackets:
    while True:
        char = brackets.read(1)
        if not char:
            break

        if char == '[':
            check.push(char)
        elif char == ']':
            if check.is_empty():
                break
            check.pop()

```

```

if check.is_empty():
    print('Скобок хватает')
else:
    print('Скобок не хватает')
#Задание 6
print("\nЗадание №6\n")
letters = Stack()
digits = Stack()
others = Stack()
result = "

```

```

with open('alf.txt', 'r') as f:
    while True:
        c = f.read(1)
        if not c:
            break
        if c.isalpha():
            letters.push(c)
        elif c.isdigit():
            digits.push(c)
        else:

```

```

        others.push(c)

letters.reverse()
digits.reverse()
others.reverse()

while not digits.is_empty():
    result += digits.pop()
while not letters.is_empty():
    result += letters.pop()
while not others.is_empty():
    result += others.pop()

print(result)
#Задание 7
print("\nЗадание №7\n")
numbers = [random.randint(-10, 10) for i in range(10)]
print(numbers)

deque = Deque()
for n in numbers:
    if n < 0:
        deque.push_left(n)
    else:
        deque.push(n)

while not deque.is_empty():
    x = deque.pop_left()
    if x < 0:
        deque.push(x)
    else:
        deque.push_left(x)
        break

while not deque.is_empty():
    x = deque.pop()
    if x < 0:
        print(x)
    else:
        deque.push(x)
        break

while not deque.is_empty():
    print(deque.pop_left())
#Задание 8

```

```
print("\nЗадание №8\n")
with open('obr.txt', 'r') as family:
    stack = Stack()
    for member in family:
        print(member)
        stack.push(member)
    print()
    while not stack.is_empty():
        print(stack.pop())
#Задание 9
print("\nЗадание №9\n")
text = 'N((TXF)AF)OT'
```

```
opstack = Stack()
vstack = Stack()
```

```
cur = 0
while True:
    read = False
    if not opstack.is_empty():
        if opstack.peek() == 'N':
            if vstack.is_empty():
                read = True
            else:
                if vstack.pop() == 'T':
                    vstack.push('F')
                else:
                    vstack.push('T')
                opstack.pop()
        elif opstack.peek() == 'A':
            if len(vstack) < 2:
                read = True
            else:
                a = vstack.pop()
                b = vstack.pop()
                if a == b and b == 'T':
                    vstack.push('T')
                else:
                    vstack.push('F')
                opstack.pop()
        elif opstack.peek() == 'O':
            if len(vstack) < 2:
                read = True
            else:
                a = vstack.pop()
```

```

        b = vstack.pop()
        if a == 'T' or b == 'T':
            vstack.push('T')
        else:
            vstack.push('F')
        opstack.pop()
    elif opstack.peek() == 'X':
        if len(vstack) < 2:
            read = True
        else:
            a = vstack.pop()
            b = vstack.pop()
            if a != b:
                vstack.push('T')
            else:
                vstack.push('F')
            opstack.pop()
    elif opstack.peek() == '(':
        read = True
    elif opstack.peek() == ')':
        opstack.pop()
        opstack.pop()
    else:
        read = True
    if read:
        i = text[cur]
        if i in 'FT':
            vstack.push(i)
        if i in 'AXON()':
            opstack.push(i)
        cur += 1

    if cur == len(text) and len(opstack) == 0:
        break

while not vstack.is_empty():
    print(vstack.pop())
#Задание 10
print("\nЗадание №10\n")
text = 'M(5, M(8, N(4, N(1, M(7, N(5, 3)))))'

op = Stack()
nums = Stack()

num = "

```

```

cur = 0
while cur < len(text):
    i = text[cur]
    if i.isdigit():
        num += i
    elif num != "":
        nums.push(int(num))
        num = ""
    if i in 'MN':
        op.push(i)
    cur += 1

```

```

while not op.is_empty():
    a = nums.pop()
    b = nums.pop()
    if a < b:
        a, b = b, a
    if op.pop() == 'M':
        nums.push(a)
    else:
        nums.push(b)

```

```

while not nums.is_empty():
    print(nums.pop())

```

#Задание 11

```
print("\nЗадание №11\n")
```

```
def check(text):
```

```
    stack = Stack()
```

```
    cur = 0
```

```
    while True:
```

```
        read = False
```

```
        if not stack.is_empty():
```

```
            if stack.peek() == '(':
```

```
                read = True
```

```
            elif stack.peek() == ')':
```

```
                stack.pop()
```

```
                if len(stack) < 2 or stack.pop() != 'formula' or stack.pop() != '(':
```

```
                    return False
```

```
                stack.push('formula')
```

```
            elif stack.peek() == 'formula':
```

```
                stack.pop()
```

```
                if len(stack) > 1 and stack.peek() in '+-':
```

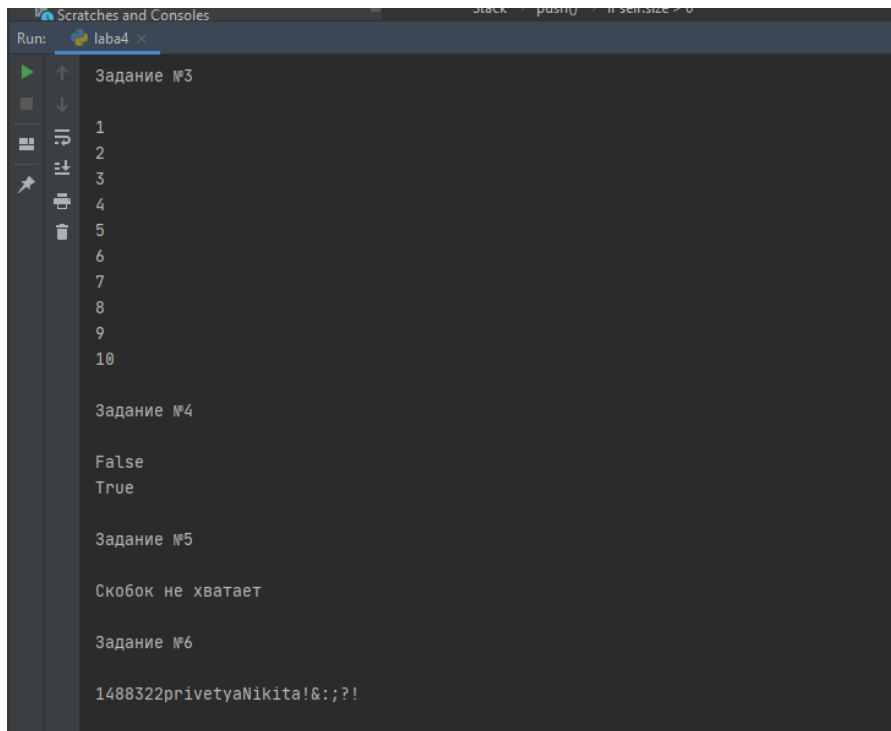



Рисунок 6 - Результат работы программы.

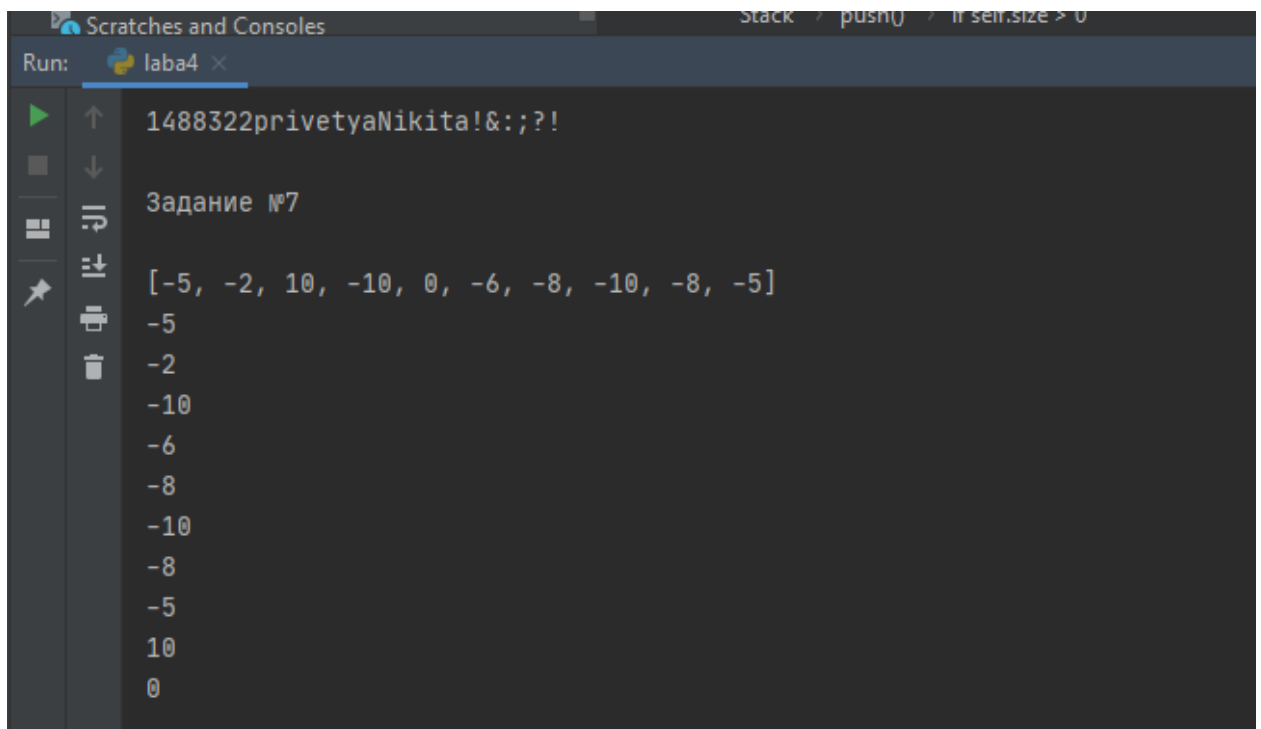


Рисунок 7 - Результат работы программы.

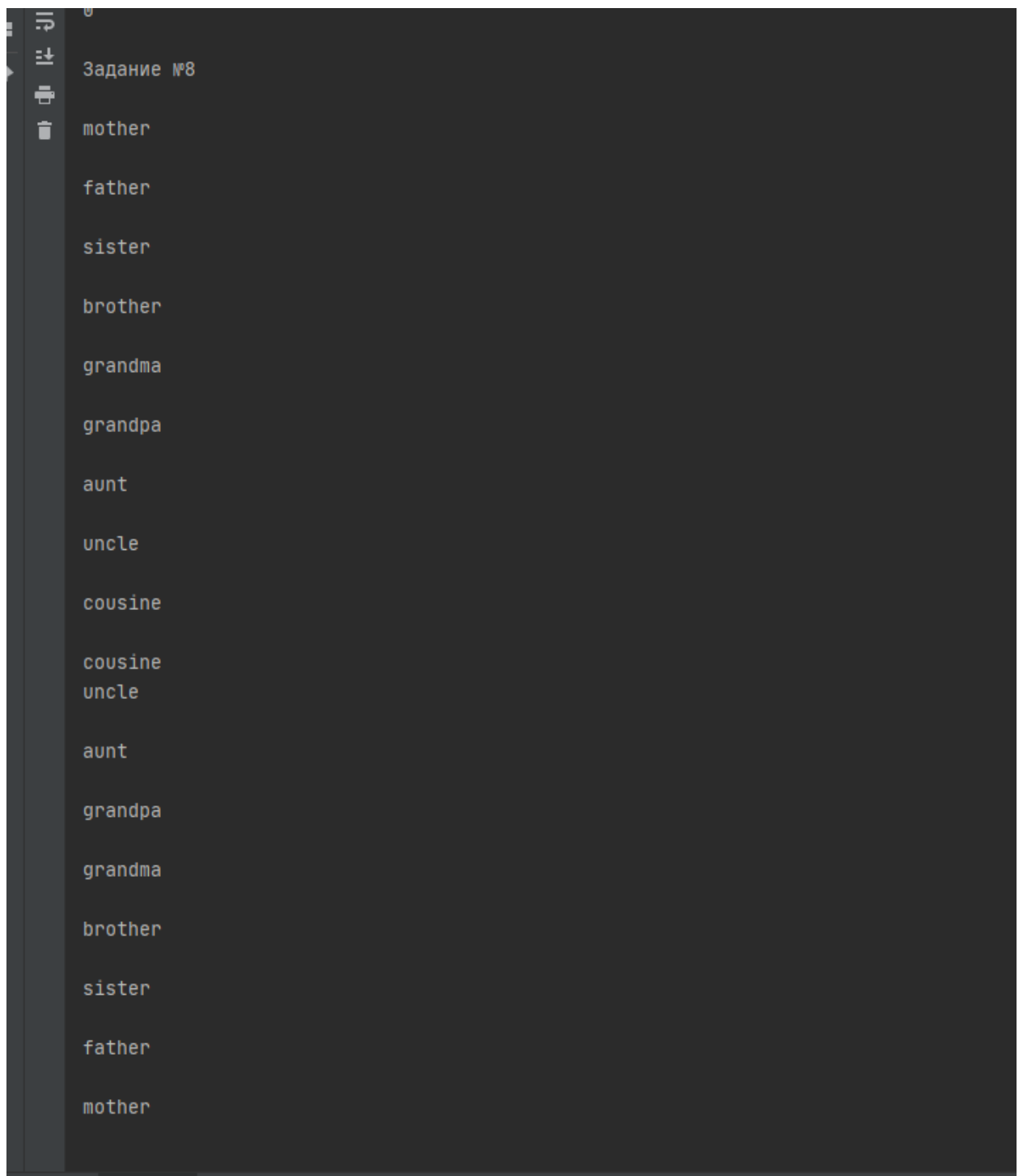


Рисунок 8 - Результат работы программы.

```
Задание №9  
T  
Задание №10  
8  
Задание №11  
False  
Process finished with exit code 0  
|
```

Рисунок 9 - Результат работы программы.

Вывод: в данной лабораторной работе была изучена теория о понятии дека и стека, а затем полученные знания были применены на практике.