САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №4
по курсу «Алгоритмы и структуры данных»
Тема: Stack Queue Linked List

Выполнил:
Лазарев Марк Олегович
К3241

Санкт-Петербург
2025 г.

Задачи по варианту

Задача №1. Очередь
Задача №2. Скобочная последовательность
Задача №3. Очередь с минимумом
Задача №4. Строй новобранцев

Задачи по варианту

2 задача. Очередь

Реализуйте работу очереди. Для каждой операции изъятия элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда — это либо «+ N», либо «–». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 10^9. Команда «–» означает изъятие элемента из очереди. Гарантируется, что размер очереди в процессе выполнения команд не превысит 10^6 элементов.

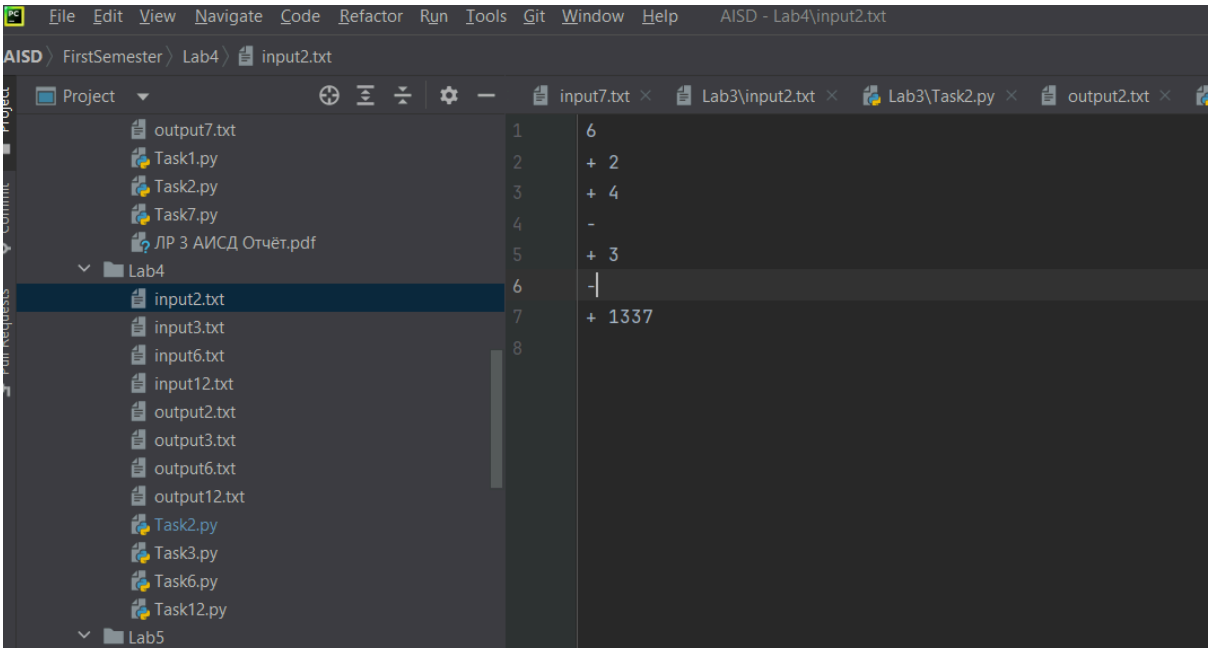
- **Формат входного файла (input.txt).** В первой строке содержится M ($1 \leq M \leq 10^6$) – число команд. В последующих строках содержатся команды, по одной в каждой строке.
- **Формат выходного файла (output.txt).** Выведите числа, которые удаляются из очереди с помощью команды «–», по одному в каждой строке. Числа нужно выводить в том порядке, в котором они были извлечены из очереди. Гарантируется, что извлечения из пустой очереди не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.

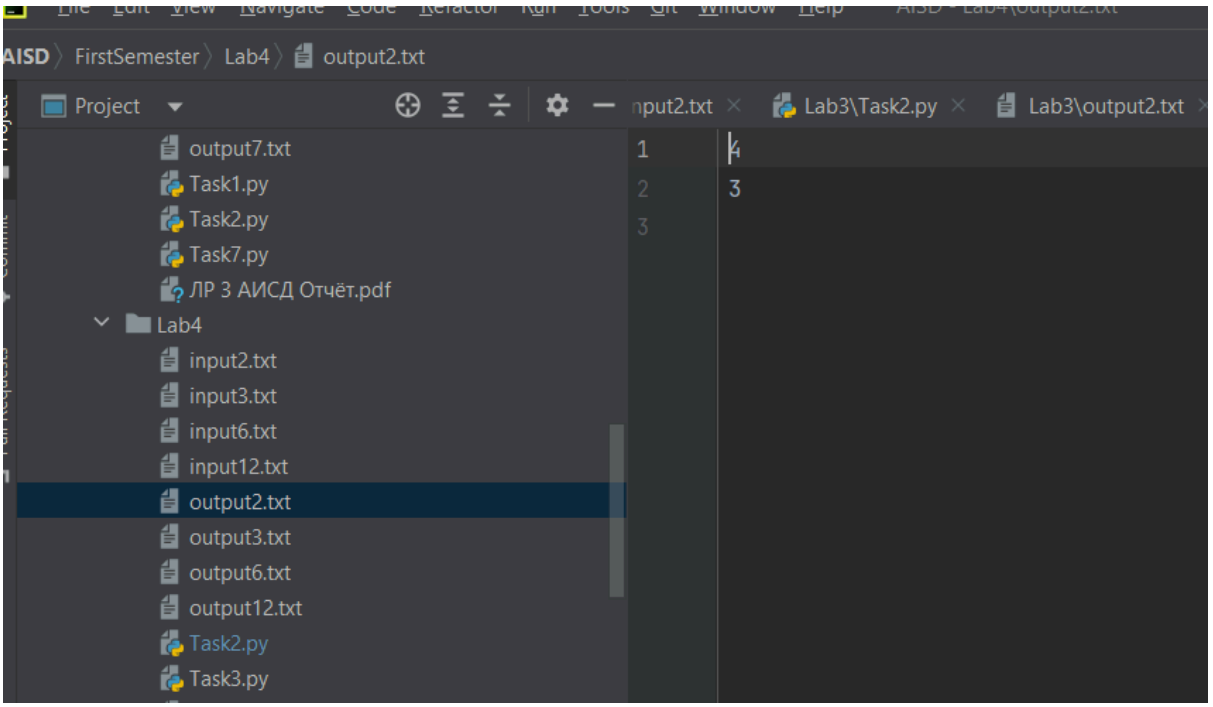
Код программы:

```
def process_stack_operations(input_file: str,
output_file: str):
    with open(input_file, "r") as infile:
```

```
        lines = infile.readlines()

    M = int(lines[0].strip())
    stack = []
    results = []

    for i in range(1, M + 1):
        command = lines[i].strip()
        if command.startswith("+"):
            _, number = command.split()
            stack.append(int(number))
        elif command == "-":
            results.append(str(stack.pop()))

    with open(output_file, "w") as outfile:
        outfile.write("\n".join(results) + "\n")

process_stack_operations("input2.txt", "output2.txt")
```

Результат работы кода на примерах:


File Edit View Navigate Code Refactor Run Tools Git Window Help
AISD - Lab4\input2.txt
AISD FirstSemester Lab4 input2.txt
Project
input7.txt
Lab3\input2.txt
Lab3\Task2.py
output2.txt
output7.txt
Task1.py
Task2.py
Task7.py
ЛР 3 АИСД Отчёт.pdf
Lab4
input2.txt
input3.txt
input6.txt
input12.txt
output2.txt
output3.txt
output6.txt
output12.txt
Task2.py
Task3.py
Task6.py
Task12.py
Lab5
6
+ 2
+ 4
-
+ 3
-
+ 1337


AISD FirstSemester Lab4 output2.txt
Project
Lab3\Task2.py
Lab3\output2.txt
output7.txt
Task1.py
Task2.py
Task7.py
ЛР 3 АИСД Отчёт.pdf
Lab4
input2.txt
input3.txt
input6.txt
input12.txt
output2.txt
output3.txt
output6.txt
output12.txt
Task2.py
Task3.py
4
3

3 задача. Скобочная последовательность. Версия 1

Последовательность A, состоящую из символов из множества «(», «)», «[» и «]», назовем ***правильной скобочной последовательностью***, если выполняется одно из следующих утверждений:

- A – пустая последовательность;
- первый символ последовательности A – это «(», и в этой последовательности существует такой символ «)», что последовательность можно представить как $A = (B)C$, где B и C – правильные скобочные последовательности;
- первый символ последовательности A – это «[», и в этой последовательности существует такой символ «]», что последовательность можно представить как $A = (B)C$, где B и C – правильные скобочные последовательности.

Так, например, последовательности «(())» и «()[]» являются правильными скобочными последовательностями, а последовательности «[)» и «((» таковыми не являются.

Входной файл содержит несколько строк, каждая из которых содержит последовательность символов «(», «)», «[» и «]». Для каждой из этих строк выясните, является ли она правильной скобочной последовательностью.

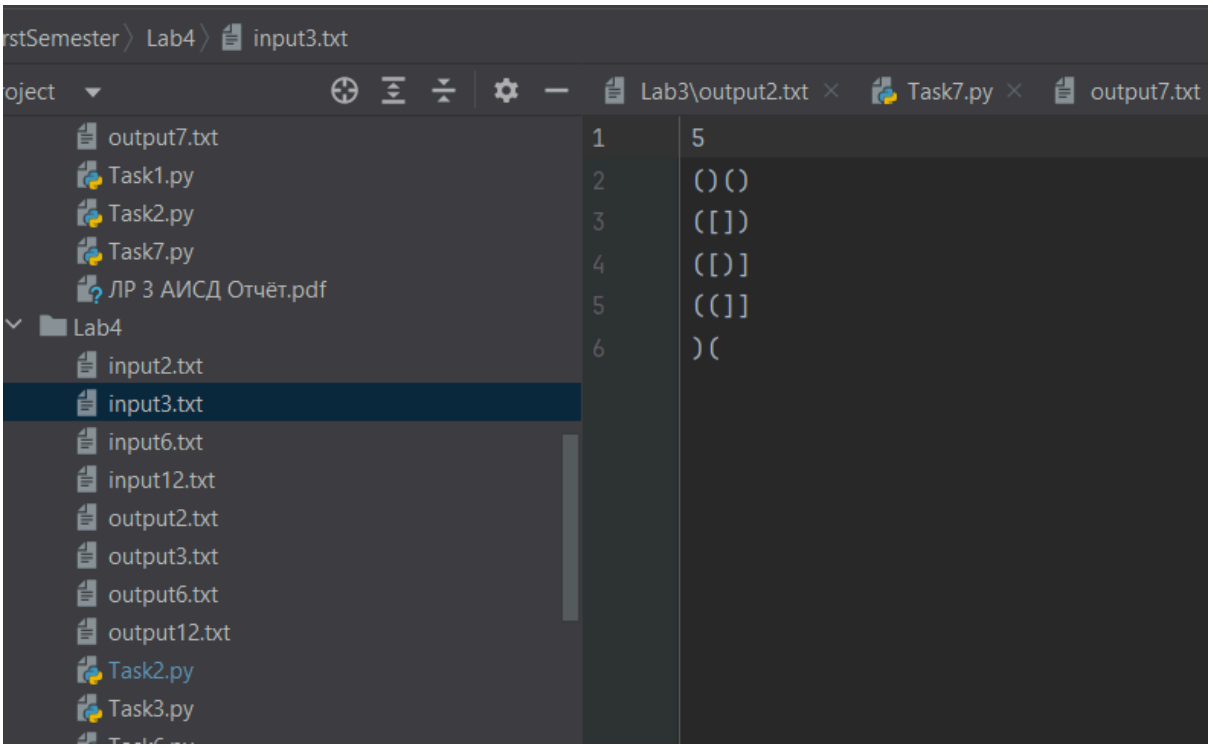
- **Формат входного файла (input.txt).** Первая строка входного файла содержит число N ($1 \leq N \leq 500$) – число скобочных последовательностей, которые необходимо проверить. Каждая из следующих N строк содержит скобочную последовательность длиной от 1 до 10^4 включительно. В каждой из последовательностей присутствуют только скобки указанных выше видов.

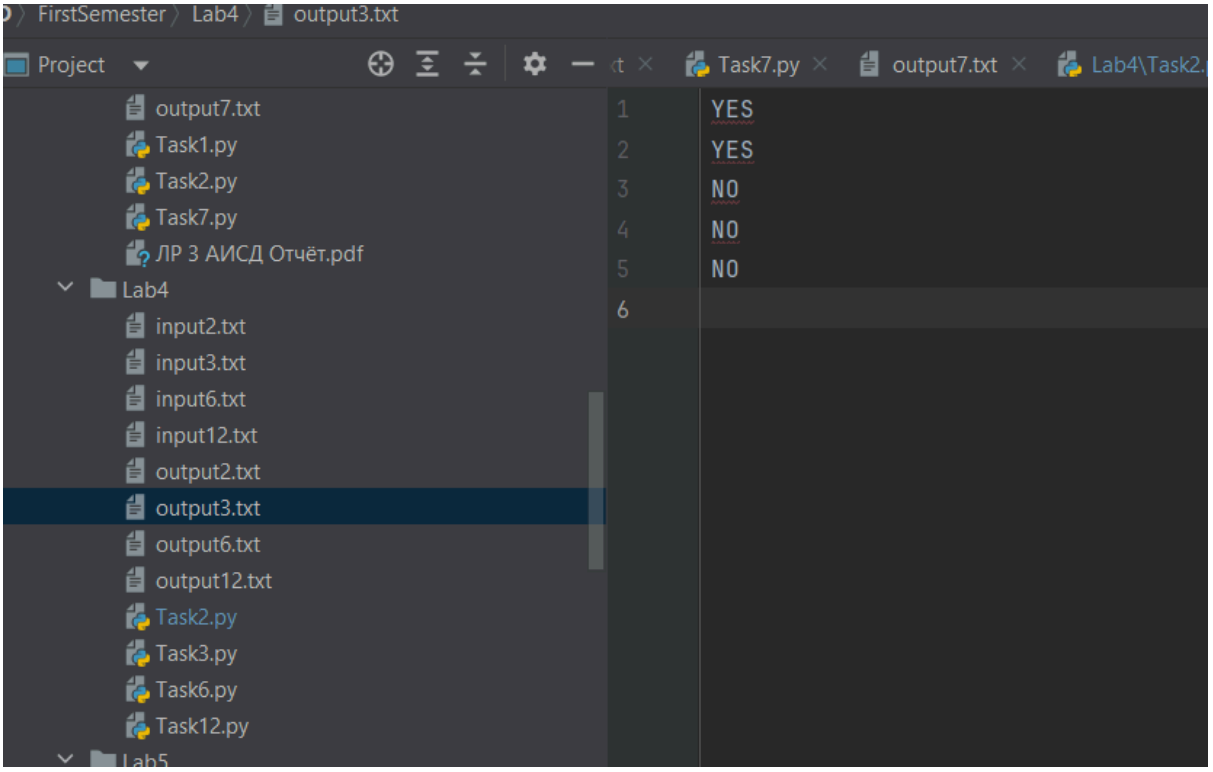
Код программы:

```
def is_correct_sequence(s):
    stack = []
    bracket_map = {')': '(', ']': '['}

    for char in s:
        if char in ('(', '['):
            stack.append(char)
        elif char in bracket_map:
                    if not stack or stack.pop() !=
bracket_map[char]:
                return False
```

```
    return not stack


with open('input3.txt', 'r') as fin,
open('output3.txt', 'w') as fout:
    n = int(fin.readline())
    for _ in range(n):
        seq = fin.readline().strip()
        if is_correct_sequence(seq):
            fout.write("YES\n")
        else:
            fout.write("NO\n")
```

Результат работы кода на примерах:




FirstSemester
Lab4
output3.txt
Project
Task7.py
output7.txt
Lab4\Task2.
output7.txt
Task1.py
Task2.py
Task7.py
ЛР 3 АИСД Отчёт.pdf
Lab4
input2.txt
input3.txt
input6.txt
input12.txt
output2.txt
output3.txt
output6.txt
output12.txt
Task2.py
Task3.py
Task6.py
Task12.py
Lab5
1 YES
2 YES
3 NO
4 NO
5 NO
6

6 задача. Очередь с минимумом

Реализуйте работу очереди. В дополнение к стандартным операциям очереди, необходимо также отвечать на запрос о минимальном элементе из тех, которые сейчас находится в очереди. Для каждой операции запроса минимального элемента выведите ее результат.

На вход программе подаются строки, содержащие команды. Каждая строка содержит одну команду. Команда – это либо «+ N», либо «–», либо «?». Команда «+ N» означает добавление в очередь числа N, по модулю не превышающего 10^9. Команда «–» означает изъятие элемента из очереди. Команда «?» означает запрос на поиск минимального элемента в очереди.

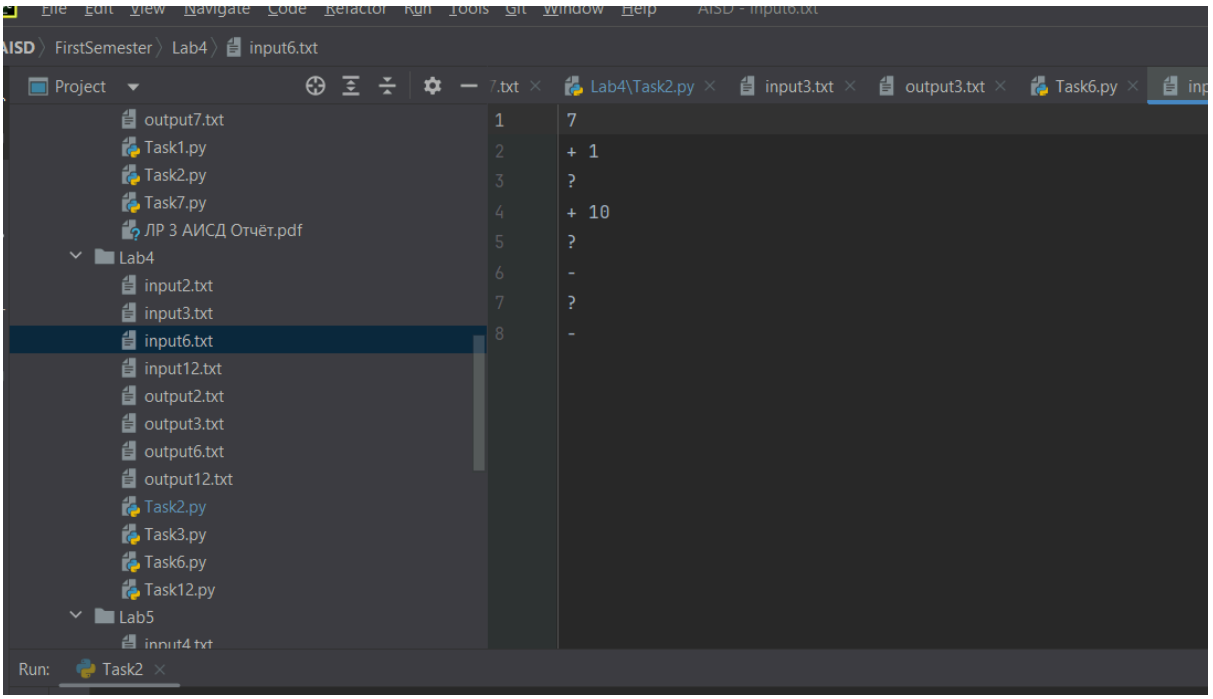
- **Формат входного файла (input.txt).** В первой строке содержится M ($1 \leq M \leq 10^6$) – число команд. В последующих строках содержатся команды, по одной в каждой строке.
- **Формат выходного файла (output.txt).** Для каждой операции поиска минимума в очереди выведите её результат. Результаты должны быть выведены в том порядке, в котором эти операции встречаются во входном файле. Гарантируется, что операций извлечения или поиска минимума для пустой очереди не производится.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

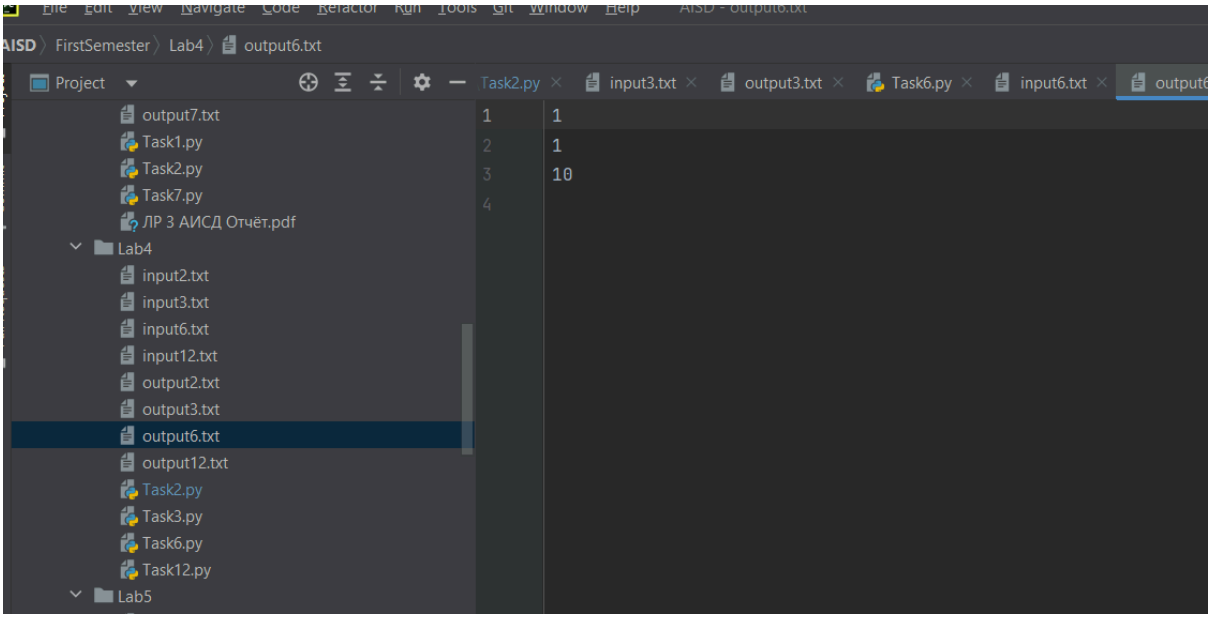
Код программы

```
from collections import deque

def process_queue(commands):
    queue = deque()
    min_queue = deque()
    results = []

    for command in commands:
        if command.startswith('+'):
            _, number = command.split()
            number = int(number)
            queue.append(number)
```

```
                while min_queue and min_queue[-1] >
number:
                min_queue.pop()
            min_queue.append(number)

        elif command == '-':
            if queue:
                removed = queue.popleft()
                if removed == min_queue[0]:
                    min_queue.popleft()

        elif command == '?':
            if min_queue:
                results.append(min_queue[0])

    return results

with open('input6.txt', 'r') as file:
    lines = file.readlines()

# Первая строка содержит количество команд
M = int(lines[0].strip())

# Остальные строки содержат команды
commands = [line.strip() for line in lines[1:M + 1]]

output = process_queue(commands)

with open('output6.txt', 'w') as file:
    for result in output:
        file.write(f"{result}\n")
```

Результат работы кода на примерах:


AISD 〉 FirstSemester 〉 Lab4 〉 input6.txt
Project
Lab4\Task2.py
input3.txt
output3.txt
Task6.py
output7.txt
Task1.py
Task2.py
Task7.py
ЛР 3 АИСД Отчёт.pdf
Lab4
input2.txt
input3.txt
input6.txt
input12.txt
output2.txt
output3.txt
output6.txt
output12.txt
Task2.py
Task3.py
Task6.py
Task12.py
Lab5
7
+ 1
?
+ 10
?
-
?
-
Run:
Task2


AISD 〉 FirstSemester 〉 Lab4 〉 output6.txt
Project
Task2.py
input3.txt
output3.txt
Task6.py
input6.txt
output7.txt
Task1.py
Task2.py
Task7.py
ЛР 3 АИСД Отчёт.pdf
Lab4
input2.txt
input3.txt
input6.txt
input12.txt
output2.txt
output3.txt
output6.txt
output12.txt
Task2.py
Task3.py
Task6.py
Task12.py
Lab5
1
1
10

12 задача. Строй новобранцев

В этой задаче n новобранцев, пронумерованных от 1 до n, разделены на два множества: *строй* и *толпа*. Вначале *строй* состоит из новобранца номер 1, все остальные составляют *толпу*. В любой момент времени строй стоит в один ряд по прямой. Товарищ сержант может использовать четыре команды. Вот они.

- "I, встать в строй слева от J."Эта команда заставляет новобранца номер I, находящегося в толпе, встать слева от новобранца номер J, находящегося встрою.
- "I, встать в строй справа от J."Эта команда действует аналогично предыдущей, за исключением того, что I встает справа от J.
- "I, выйти из строя."Эта команда заставляет выйти из строя новобранца номер I. После этого он присоединяется ктолпе.
- "I, назвать соседей."Эта команда заставляет глубоко задуматься новобранца номер I, стоящего в строю, и назвать номера своих соседей по строю, сначала левого, потом правого. Если кто-то из них отсутствует (новобранец находится на краю ряда), то вместо соответствующего номера он должен назвать 0.

Известно, что ни в каком случае строй не остается пустым. Иногда строй становится слишком большим, и товарищ сержант уже не может проверять сам, правильно ли отвечает новобранец. Поэтому он попросил вас написать программу, которая помогает ему в нелегком деле обучения молодежи и выдает правильные ответы для его команд.

- **Формат входного файла (input.txt).** В первой строке находятся два числа N ($1 \leq N \leq 75000$) и M ($1 \leq M \leq 75000$) - количество новобранцев и команд соответственно. Следующие M строк содержат команды, одна команда на строку. Каждая команда - одна из следующих:
  - `left I J` - соответствует команде "I, встать в строй слева от J."
  - `right I J` - "I, встать в строй справа от J."
  - `leave I` - "I, выйти из строя."
  - `name I` - "I, назвать соседей."

  Гарантируется, что все команды корректны, например, `leave I` не будет заставлять выйти из строя новобранца, стоящего в толпе. Также гарантируется, что строй никогда не будет пустым.

```
import sys
from collections import OrderedDict

with open("input12.txt", "r") as file:
    n, m = map(int, file.readline().split())
```

```
    commands = [line.strip().split() for line in
file]

# OrderedDict для хранения порядка в строю
recruits = OrderedDict()
recruits[1] = [None, None]

output = []

for command in commands:
    if command[0] == "left":
        i, j = map(int, command[1:])
        left, right = recruits[j][0], j
        recruits[i] = [left, right]
        recruits[j][0] = i
        if left is not None:
            recruits[left][1] = i
    elif command[0] == "right":
        i, j = map(int, command[1:])
        left, right = j, recruits[j][1]
        recruits[i] = [left, right]
        recruits[j][1] = i
        if right is not None:
            recruits[right][0] = i
    elif command[0] == "leave":
        i = int(command[1])
        left, right = recruits[i]
        if left is not None:
            recruits[left][1] = right
        if right is not None:
            recruits[right][0] = left
        del recruits[i]
    elif command[0] == "name":
        i = int(command[1])
        left, right = recruits[i]
        output.append(f"{left or 0} {right or 0}")

with open("output12.txt", "w") as file:
```

```
    file.write("\n".join(output) + "\n")
```

Результат работы кода на примерах: