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

Отчет по лабораторной работе №6

по курсу «Алгоритмы и структуры данных»

Тема: Hashing

Выполнил:
Лазарев Марк Олегович
К3241

Санкт-Петербург
2025 г.

Задачи по варианту

Задача №1. Множество

Задача №2. Прошитый ассоциативный массив

Задача №3. Выборы в США

Задачи по варианту

1 задача. Множество

Реализуйте множество с операциями «добавление ключа», «удаление ключа», «проверка существования ключа».

- **Формат входного файла (input.txt).** В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 10^5$. В каждой из последующих N строк находится одна из следующих операций:
  - A x – добавить элемент x в множество. Если элемент уже есть в множестве, то ничего делать не надо.
  - D x – удалить элемент x. Если элемента x нет, то ничего делать не надо.
  - ? x – если ключ x есть в множестве, выведите «Y», если нет, то выведите «N».

  Аргументы указанных выше операций – **целые числа**, не превышающие по модулю 10^{18}.
- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций «?». Следуйте формату выходного файла из примера.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 256 мб.
- Пример:

input.txt	output.txt
8	Y
A 2	N
A 5	N
A 3	
? 2	
? 4	
A 2	
D 2	
? 2	

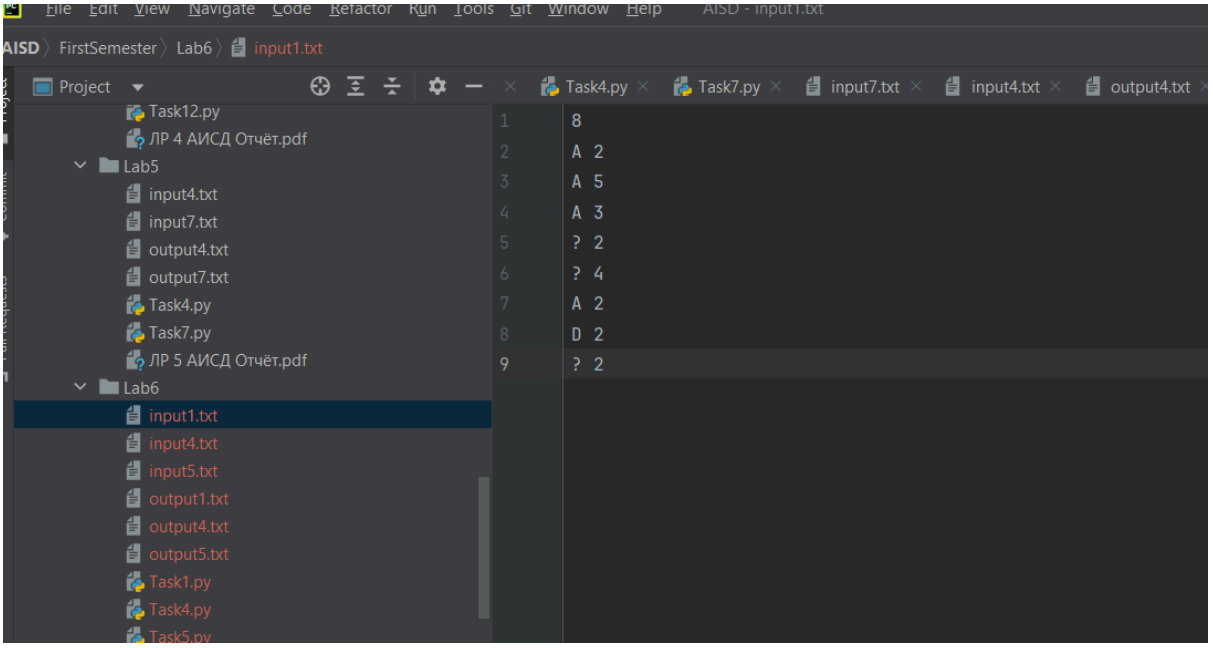
Код программы:

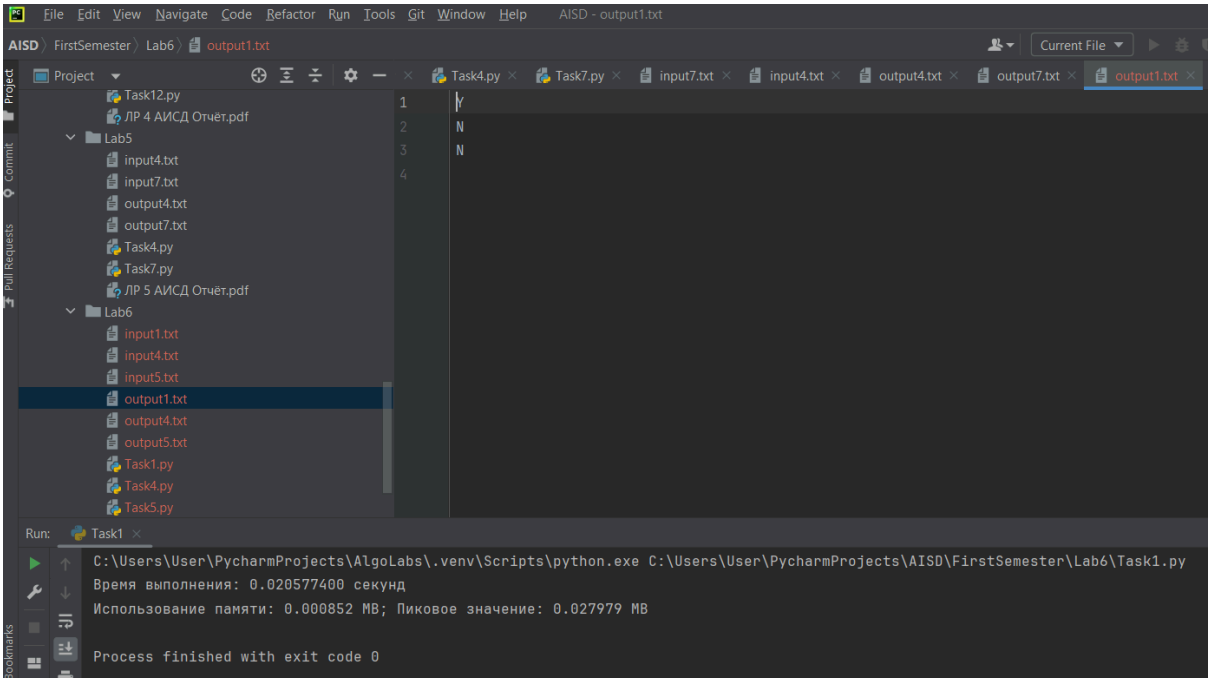
```
import time
import tracemalloc
```

```
def process_operations():
        with   open('input1.txt',   'r')   as   infile,
open('output1.txt', 'w') as outfile:
       n = int(infile.readline().strip())
       my_set = set()

       for _ in range(n):
                                          operation    =
infile.readline().strip().split()
           op_type = operation[0]
           if op_type == 'A':
               x = int(operation[1])
               my_set.add(x)
           elif op_type == 'D':
               x = int(operation[1])
               my_set.discard(x)
           elif op_type == '?':
               x = int(operation[1])
               if x in my_set:
                   outfile.write('Y\n')
               else:
                   outfile.write('N\n')


def main():
   start_time = time.perf_counter()
   tracemalloc.start()
   process_operations()
   end_time = time.perf_counter()
   current, peak = tracemalloc.get_traced_memory()
   tracemalloc.stop()

         print(f"Время   выполнения:   {end_time   -
start_time:.9f} секунд")
     print(f"Использование  памяти:  {current  /  10  **
6:.6f}  MB;  Пиковое  значение:  {peak  /  10  **  6:.6f}
MB")
```

```
if __name__ == "__main__":
    main()
```

Результат работы кода на примерах:





4 задача. Прошитый ассоциативный массив

Реализуйте прошитый ассоциативный массив. Ваш алгоритм должен поддерживать следующие типы операций:

- `get` x – если ключ x есть в множестве, выведите соответствующее ему значение, если нет, то выведите <none>.
- `prev` x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен позже всех, но до x, или <none>,

если такого нет или в массиве нет x.

- `next` x – вывести значение, соответствующее ключу, находящемуся в ассоциативном массиве, который был вставлен раньше всех, но после x , или <none>, если такого нет или в массиве нет x .
- `put` x y – поставить в соответствие ключу x значение y. При этом следует учесть, что
  - если, независимо от предыстории, этого ключа на момент вставки в массиве не было, то он считается только что вставленным и оказывается самым последним среди добавленных элементов – то есть, вызов `next` с этим же ключом сразу после выполнения текущей операции put должен вернуть <none>;
  - если этот ключ уже есть в массиве, то значение необходимо изменить, и в этом случае ключ не считается вставленным еще раз, то есть, не меняет своего положения в порядке добавленных элементов.
- `delete` x – удалить ключ x. Если ключа в ассоциативном массиве нет, то ничего делать не надо.
- **Формат входного файла (input.txt).** В первой строке входного файла находится строго положительное целое число операций N, не превышающее $5 \cdot 10^5$. В каждой из последующих N строк находится одна из приведенных выше операций. Ключи и значения операций - строки из латинских букв длиной не менее одного и не более 20 символов.
- **Формат выходного файла (output.txt).** Выведите последовательно результат выполнения всех операций get, prev, next. Следуйте формату выходного файла из примера.
- Ограничение по времени. 4 сек.
- Ограничение по памяти. 256 мб.
- Примеры:

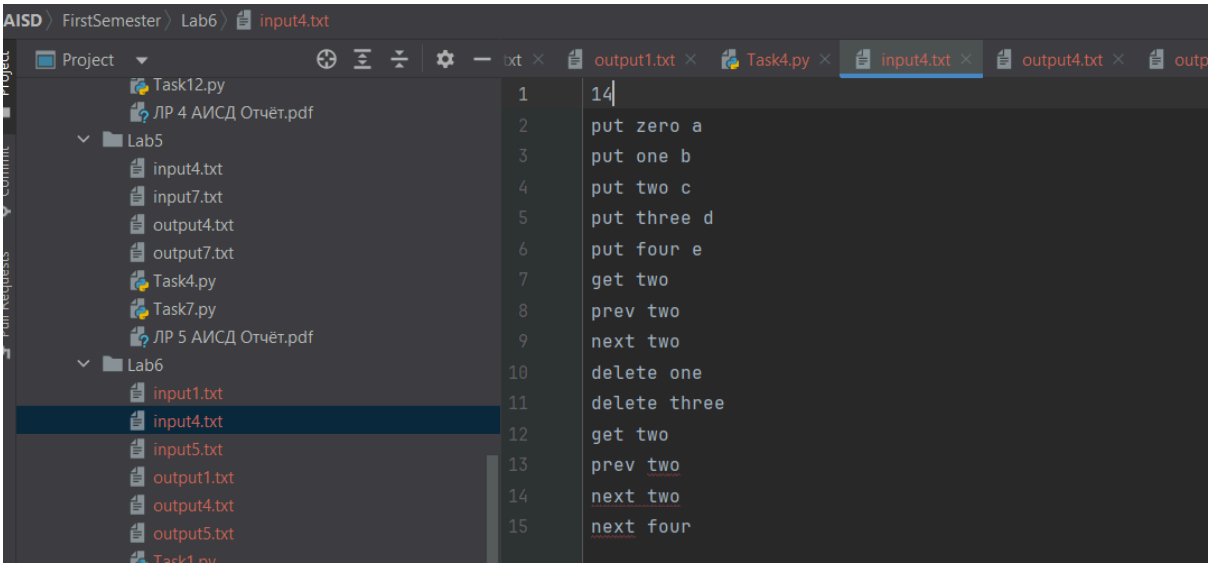
Код программы:

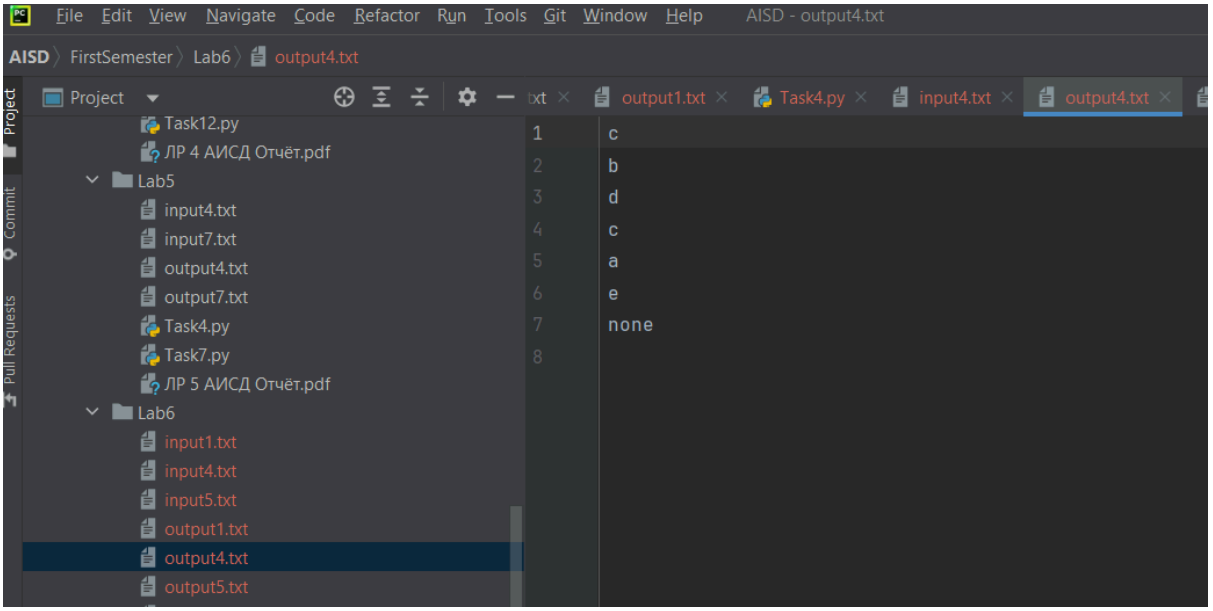
```
class ThreadedMap:
    def __init__(self):
                # Инициализация словаря для хранения
ключ-значение и списка для порядка вставки
        self.map = {}
        self.order = []

    def put(self, x, y):
        # Добавление или обновление значения
        if x not in self.map:
            self.order.append(x)
        self.map[x] = y

    def delete(self, x):
        # Удаление ключа, если он существует
        if x in self.map:
            del self.map[x]
            self.order.remove(x)

    def get(self, x):
            # Получение значения по ключу или 'none',
если ключа нет
        return self.map.get(x, 'none')

    def prev(self, x):
            # Получение предыдущего значения в порядке
вставки
        if x not in self.map:
            return 'none'
        index = self.order.index(x)
        if index == 0:
            return 'none'
        return self.map[self.order[index - 1]]

    def next(self, x):
```

```
        # Получение следующего значения в порядке вставки
        if x not in self.map:
            return 'none'
        index = self.order.index(x)
        if index == len(self.order) - 1:
            return 'none'
        return self.map[self.order[index + 1]]


def process_operations(input_file, output_file):
    # Обработка операций из входного файла и запись результатов в выходной файл
    threaded_map = ThreadedMap()
        with open(input_file, 'r') as infile, open(output_file, 'w') as outfile:
        n = int(infile.readline().strip())

        for _ in range(n):
                                        operation = infile.readline().strip().split()
            op_type = operation[0]
            x = operation[1]

            if op_type == 'put':
                y = operation[2]
                threaded_map.put(x, y)
            elif op_type == 'delete':
                threaded_map.delete(x)
            elif op_type == 'get':
                result = threaded_map.get(x)
                outfile.write(f"{result}\n")
            elif op_type == 'prev':
                result = threaded_map.prev(x)
                outfile.write(f"{result}\n")
            elif op_type == 'next':
                result = threaded_map.next(x)
                outfile.write(f"{result}\n")
```

```
def main():
    # Основная функция для запуска обработки операций
    process_operations('input4.txt', 'output4.txt')


if __name__ == "__main__":
    main()
```

Результат работы кода на примерах:





5 задача. Выборы в США

Как известно, в США президент выбирается не прямым голосованием, а путем двухуровневого голосования. Сначала проводятся выборы в каждом штате и определяется победитель выборов в данном штате. Затем проводятся государственные выборы: на этих выборах каждый штат имеет определенное число голосов — число выборщиков от этого штата. На практике, все выборщики от штата голосуют в соответствии с результатами голосования внутри штата, то есть на заключительной стадии выборов в голосовании участвуют штаты, имеющие различное число голосов. Вам известно за кого проголосовал каждый штат и сколько голосов было отдано данным штатом. Подведите итоги выборов: для каждого из участника голосования определите число отданных за него голосов.

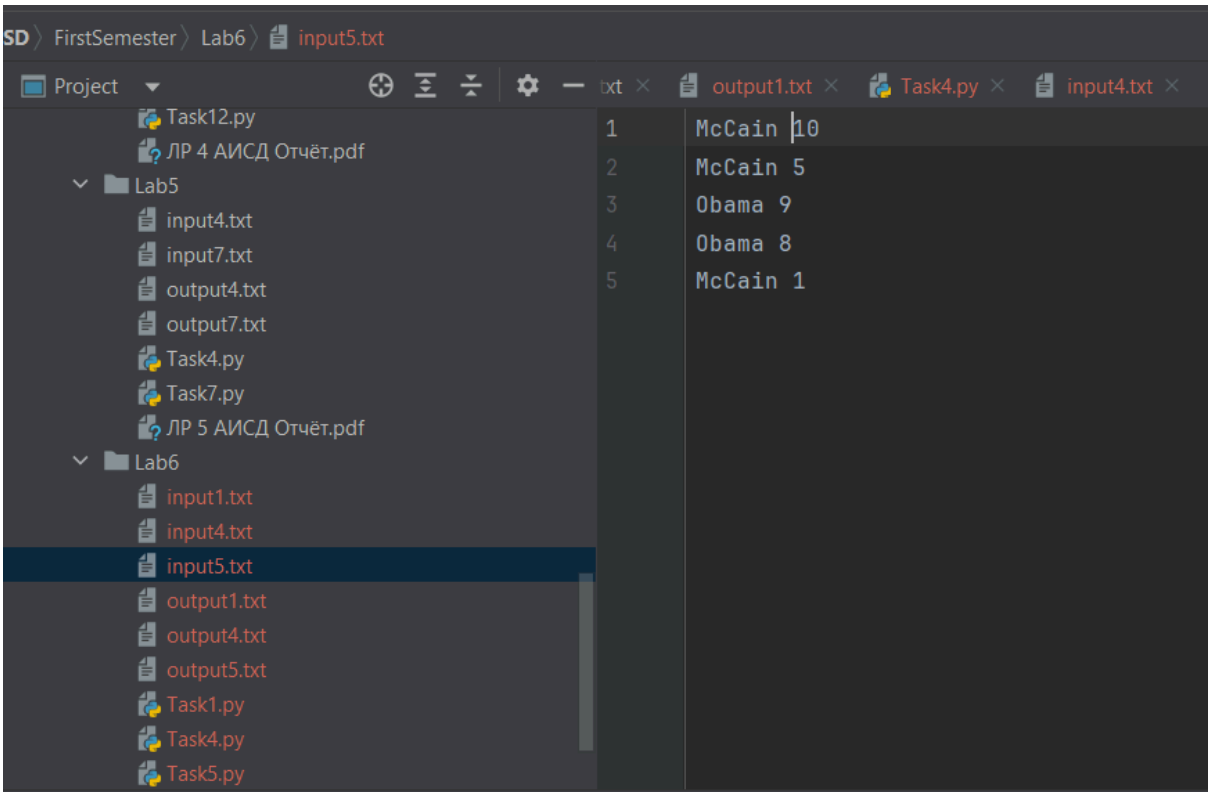
- **Формат ввода / входного файла (input.txt).** Каждая строка входного файла содержит фамилию кандидата, за которого отдают голоса выборщики этого штата, затем через пробел идет количество выборщиков, отдавших голоса за этого кандидата.
- **Формат вывода / выходного файла (output.txt).** Выведите фамилии всех кандидатов в ***лексикографическом*** порядке, затем, через пробел, количество отданных за них голосов.
- Ограничение по времени. 2 сек.
- Ограничение по памяти. 64 мб.
- Примеры:

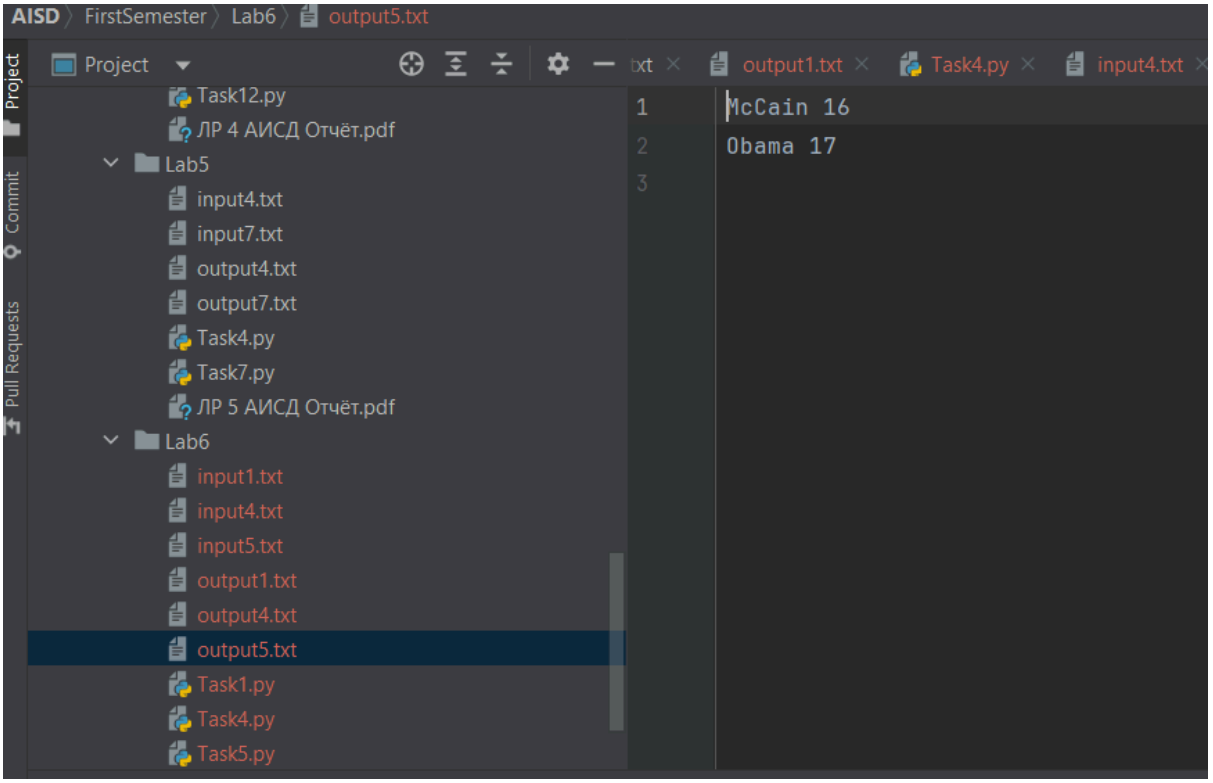
Код программы:

```
def process_votes(input_file, output_file):
    votes = {}

    with open(input_file, 'r') as infile:
        for line in infile:
            candidate, count = line.strip().split()
            count = int(count)
            if candidate in votes:
                votes[candidate] += count
            else:
                votes[candidate] = count

    with open(output_file, 'w') as outfile:
```

```
        for candidate in sorted(votes.keys()):
                                outfile.write(f"{candidate}
{votes[candidate]}\n")

def main():
    process_votes('input5.txt', 'output5.txt')

if __name__ == "__main__":
    main()
```

Результат работы кода на примерах:




AISD
FirstSemester
Lab6
output5.txt
Project
Commit
Pull Requests
output1.txt
Task4.py
input4.txt
Task12.py
ЛР 4 АИСД Отчёт.pdf
Lab5
input4.txt
input7.txt
output4.txt
output7.txt
Task4.py
Task7.py
ЛР 5 АИСД Отчёт.pdf
Lab6
input1.txt
input4.txt
input5.txt
output1.txt
output4.txt
output5.txt
Task1.py
Task4.py
Task5.py
1 McCain 16
2 Obama 17
3