МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №2
по дисциплине «Информатика»
Тема: Моделирование работы Машины Тьюринга

Студент гр. 9383 _____ Ноздрин В.Я.

Преподаватель _____ Размочаева Н.В.

Санкт-Петербург
2019

Цель работы.

Целью данной работы является моделирование работы машины Тьюринга на языке **Python**.

Задание.

На вход программе подается строка неизвестной длины. Каждый элемент является значением в ячейке памяти ленты Машины Тьюринга.

На ленте находится троичное число, знак (плюс или минус) и троичная цифра.

Необходимо написать программу, которая выполнит арифметическую операцию. Указатель на текущее состояние Машины Тьюринга изначально находится слева от числа (но не на первом его символе). По обе стороны от числа находятся пробелы. Результат арифметической операции запишите на месте первого числа. Для примера выше лента будет выглядеть так:

Ваша программа должна вывести полученную ленту после завершения работы.

Алфавит: '0'; '1'; '2'; '+'; '-'; ' '(пробел),

Соглашения:

1. Направление движения автомата должно быть: R (направо), L (налево) или N (неподвижно).

2. Число обязательно начинается с единицы или двойки.

3. Числа и знак операции между ними идут непрерывно.

4. Гарантируется, что в результате операции вычитания не может получиться отрицательного числа.

Выполнение работы.

Переменные:

states - Словарь, таблица значений машины Тьюринга

line - Лента машины Тьюринга, пользовательский ввода

state - переменная, хранящая текущее состояние машины Тьюринга

value - значение текущей ячейки ленты машины Тьюринга

new_value, next_state - значение, которое необходимо записать на ленту и следующее состояние, в которое перейдет машина Тьюринга

position - текущее положение автомата

move - направление движения автомата, сдвиг

Функции:

gen_dic() - функция генерирует словарь ***states***, как строку и печатает ее в стандартный поток вывода. В ходе работы программы не используется, но использовалась, чтобы создавать сам код программы.

Описание состояний машины Тьюринга :

q1 - начальное состояние, в котором автомат проходит повсей строки до знака '+' или '-', чтобы определить действие: сложение или вычитание.

q2 - в это состояние автомат переходит в случае выполнения операции сложения. Выбирает следующее состояние в зависимости от второго аргумента.

q3 - в это состояние автомат переходит в случае, если необходимо прибавить 1.

q4 - в это состояние автомат переходит в случае, если необходимо прибавить 2.

q5 - в это состояние автомат переходит в случае выполнения операции вычитания.

q6 - в это состояние автомат переходит в случае, если необходимо вычесть 1.

q7 - в это состояние автомат переходит в случае, если необходимо вычесть 2.

q8 - в это состояние автомат переходит в случае, если необходимо прибавить или вычесть 0. По условию второй аргумент не 0, так что это состояние бесполезное.

q9 - в это состояние автомат переходит в случае, если при выполнении какой-либо операции ячейка заменена на '0' и операция завершилась. Данное состояние нужно для удаления незначащего нуля с помощью состояния q10

q10 - в это состояние автомат переходит в случае необходимости удалить незначащий ноль. Удаляет ноль и проверяет, есть ли справа знак '+' или '-'. Если есть, переходит в q11, чтобы вернуть 0.

q11 - Возвращает 0, если в результате вычитания получается 0.

qn - терминальное состояние, необходимо только для того, чтобы обозначить завершение работы машины Тьюринга.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица – Результаты тестирования

№ п/п	Входные данные	Выходные данные
1	11-2	2-2
2	100-1	22-1

5 Выводы.

Были изучены основные понятия, связанный с машиной Тьюринга.

Была разработана программа, моделирующая работу машины Тьюринга. Для хранения таблицы значений использовался словарь словарей.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.py

```
def gen_dic(n):
    print("states = {", end='')
    for i in range(n):
        print(
            """
    'q{}': {{
        '0': ('0', 0, 'q{}'),
        '1': ('1', 0, 'q{}'),
        '2': ('2', 0, 'q{}'),
        ' ': (' ', 0, 'q{}'),
        '+': ('+', 0, 'q{}'),
        '-': ('-', 0, 'q{}'),
    }},""".format(i, i, i, i, i, i)
        )
    print("}")



states = {
    # reading until '+' or '-'
    'q1': {
        '0': ('0', 1, 'q1'),
        '1': ('1', 1, 'q1'),
        '2': ('2', 1, 'q1'),
        ' ': (' ', 1, 'q1'),
        '+': ('+', 1, 'q2'),
        '-': ('-', 1, 'q5'),
    },


    # reading second arg for sum
    'q2': {
        '0': ('0', -1, 'q8'),
        '1': ('1', -1, 'q3'),
        '2': ('2', -1, 'q4'),
    },
    # +1
    'q3': {
        '0': ('1', 0, 'qn'),
```

```
    '1': ('2', 0, 'qn'),
    '2': ('0', -1, 'q3'),
    ' ': ('1', 0, 'qn'),
    '+': ('+', -1, 'q3'),
},
# +2
'q4': {
    '0': ('2', 0, 'qn'),
    '1': ('0', -1, 'q3'),
    '2': ('1', -1, 'q3'),
    ' ': ('2', 0, 'qn'),
    '+': ('+', -1, 'q4'),
},

# reading second arg for dif
'q5': {
    '0': ('0', -1, 'q8'),
    '1': ('1', -1, 'q6'),
    '2': ('2', -1, 'q7'),
},
# -1
'q6': {
    '0': ('2', -1, 'q6'),
    '1': ('0', -1, 'q9'),
    '2': ('1', 0, 'qn'),
    '-': ('-', -1, 'q6'),
},
# -2
'q7': {
    '0': ('1', -1, 'q6'),
    '1': ('2', -1, 'q6'),
    '2': ('0', -1, 'q9'),
    '-': ('-', -1, 'q7'),
},

# remove +- if arg is 0
# +-0
'q8': {
    '+': ('+', 0, 'qn'),
    '-': ('-', 0, 'qn'),
},
```

```
    # delete 0
    'q9': {
        '0': ('0', 0, 'qn'),
        '1': ('1', 0, 'qn'),
        '2': ('2', 0, 'qn'),
        ' ': (' ', +1, 'q10'),
    },
    'q10': {
        '0': (' ', 1, 'q10'),
        '1': ('1', 0, 'qn'),
        '2': ('2', 0, 'qn'),
        '+': ('+', -1, 'q11'),
        '-': ('-', -1, 'q11'),
    },
    'q11': {
        ' ': ('0', 0, 'qn'),
    },

    # terminal state
    'qn': {}
}


def main():
    line = list(' '*2 + input() + ' '*2)
    position = 0
    state = 'q1'
    while state != 'qn':
        value = line[position]
        new_value, move, next_state = states[state][value]
        line[position], state = new_value, next_state
        position += move
        if position < 0:
            raise IndexError('string index out of range')
    print(''.join(line[2:-2]))


main()
```