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

ОТЧЕТ

по лабораторной работе №3

по дисциплине « Информатика »

Тема: Машина Тьюринга.

Студентка гр. 2384 _____ Валеева А. А.

Преподаватель _____ Шевская Н. В.

Санкт-Петербург

2022

Цель работы.

Целью данной лабораторной работы является изучение работы Машины Тьюринга и разработка программы для неё.

Задание.

Вариант 1

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

На ленте находится последовательность латинских букв из алфавита {a, b, c}.

			a	c	c	a	b	c	b	a	b

Напишите программу, которая удаляет в исходной строке два символа, следующих за первым встретившимся символом ‘b’. Если первый встретившийся символ ‘b’ – последний в строке, то удалить его. Если первый встретившийся символ ‘b’ – предпоследний в строке, то удалить один символ, следующий за ним, т. е. последний в строке. Если в строке символ ‘b’ отсутствует, то удалить самый первый символ строки. После удаления в строке не должно оставаться пробелов и пустых мест!

Указатель на текущее состояние Машины Тьюринга изначально находится слева от строки с символами (но не на первом ее символе). По обе стороны от строки находятся пробелы.

Для примера выше лента будет выглядеть так:

			a	c	c	a	b	a	b	a	a

Алфавит:

- a
- b
- c
- " " (пробел)

Соглашения:

1. Направление движения автомата может быть одно из R (направо), L (налево), N (неподвижно).

2. Гарантируется, что длинна строки не менее 5 символов и не более 13.

3. В середине строки не могут встретиться пробелы.

4. При удалении или вставке символов направление сдвигов подстрок не принципиально (т. е. результат работы алгоритма может быть сдвинут по ленте в любую ее сторону на любое число символов).

5. Курсор по окончании работы алгоритма может находиться на любом символе.

Ваша программа должна вывести полученную ленту после завершения работы.

В отчете предоставьте таблицу состояний. Отдельно кратко опишите каждое состояние, например:

q1 - начальное состояние, которое необходимо, чтобы найти первый встретившийся символ ‘b’.

Выполнение работы.

Начальное сосятоние: q_start.

Далее в q0 проверяется наличие символа b, если его нет — тогда переходим в состояния q_find_1 и q_del_1 для удаления первого символа слова.

Частные случаи (символ b) последний или предпоследний рассмотрены в состоянии q_special. После символа b следующие два символа (которые следует удалить) зануляем и смотрим следующие символы. Если находим символ b, то зануляем и переходим в состояние qbb, от которого в q_replace_b, где меняем первый ноль на b. То же самое проделываем, если встречен символ a или c (qaa + q_replace_a или qcc+q_replace_c). С помощью состония q5 переносим символы вправо на место нулей. В конце меняем нули на пробелы с помощью состояния q_replace_0, Конечное состояние q_end.

Разработанный программный код см. в приложении А.

Тестирование.

Результаты тестирования представлены в табл. 1.

Таблица 1 – Результаты тестирования

№ п/п	Входные данные	Выходные данные	Комментарии
1.	aaa	aa	Отсутствие буквы b.
2.	aabbbcacacacacc	aabcacacacacc	Удаление из середины предложения.
3.	aab	aa	Буква b в конце слова.
4.	aaba	aab	Буква b предпоследняя.

Выводы.

На практическом примере была изучена Машина Тьюринга, а также написана программа для неё.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: Valeeva_Alina_lb3.py

```
R, N, L = 1, 0, -1
table = {
    # начальное состояние
    'q_start': {
        ' ': [' ', R, 'q_start'],
        'a': ['a', N, 'q0'],
        'c': ['c', N, 'q0'],
        'b': ['b', N, 'q0']
    },
    # проверяется, есть ли символ b
    'q0':{
        ' ': [' ', L, 'q_find_1' ], # прописать удаление первого
символа
        'a': ['a', R, 'q0'],
        'c': ['c', R, 'q0'],
        'b': ['b', R, 'q1']
    },
    # поиск первого символа (если нет b)
    'q_find_1': {
        'a': ['a', L, 'q_find_1'],
        'b': ['b', L, 'q_find_1'],
        'c': ['c', L, 'q_find_1'],
        ' ': [' ', R, 'q_del_1']
    },
    # удаление первого символа (если нет b)
    'q_del_1': {
        'a': [' ', N, 'q_end'],
        'b': [' ', N, 'q_end'],
        'c': [' ', N, 'q_end']
    },
    # что делать, когда встретили символ b
    'q1': {
        'a': ['0', R, 'q2'],
        'b': ['0', R, 'q2'],
        'c': ['0', R, 'q2'],
        ' ': [' ', L, 'q_special']
    },
    # второй символ после b
    'q2': {
        'a': ['0', R, 'q3'],
        'b': ['0', R, 'q3'],
        'c': ['0', R, 'q3'],
        ' ': [' ', L, 'q_special']
    },
    # частные случаи расположения b
    'q_special': {
        '0': [' ', N, 'q_end'], # если символ b предпоследний
        'b': [' ', N, 'q_end'], # если символ b последний
    },
    # нашли символ после двух удаляемых
    'q3': {
        'b': ['0', L, 'qbb'],
        'a': ['0', L, 'qaa'],
```

```
            'c': ['0', L, 'qcc'],
            ' ': [' ', L, 'q_replace_0']
        },
        # поиск символов после двух удаляемых
        'q5': {
            '0': ['0', R, 'q5'],
            'b': ['0', L, 'qbb'],
            'a': ['0', L, 'qaa'],
            'c': ['0', L, 'qcc'],
            ' ': [' ', L, 'q_replace_0']
        },
        # символы, замененные на 0 удаляем, пока не дойдем до конца
слова
        'q_replace_0': {
            '0': [' ', L, 'q_replace_0'],
            'a': ['a', N, 'q_end'],
            'b': ['b', N, 'q_end'],
            'c': ['c', N, 'q_end']
        },
        # если нашли b
        'qbb':{
            '0': ['0', L, 'qbb'],
            'b': ['b', R, 'q_replace_b'],
            'a': ['a', R, 'q_replace_b'],
            'c': ['c', R, 'q_replace_b']
        },
        # замена на b
        'q_replace_b': {
            '0': ['b', R, 'q5']
        },
        # если нашли c
        'qcc': {
            '0': ['0', L, 'qcc'],
            'a': ['a', R, 'q_replace_c'],
            'b': ['b', R, 'q_replace_c'],
            'c': ['c', R, 'q_replace_c']
        },
        # замена на c
        'q_replace_c':{
            '0': ['c', R, 'q5']
        },
        # если нашли a
        'qaa': {
            '0': ['0', L, 'qaa'],
            'a': ['a', R, 'q_replace_a'],
            'b': ['b', R, 'q_replace_a'],
            'c': ['c', R, 'q_replace_a']
        },
        # замена на a
        'q_replace_a': {
            '0': ['a', R, 'q5']
        }
    }

    word = list(input())
    state = 'q_start'
    index = 0
    while state != 'q_end':
        symbol, i, state = table[state][word[index]]
        word[index] = symbol
```

```
            index += i
        print(*word, sep = '')
```

Пример и состояния:

До: aabcabc

После: aabbc

Состояния: q_start q0 q0 q0 q1 q2 q3 qbb qbb qbb q_replace_b q5 q5 q5 qcc qcc qcc q_replace_c q5 q5 q5 q_replace_0 q_replace_0 q_replace_0 q_end