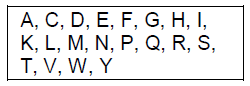
**Строки, структуры, массивы, коллекции.**

**ГЕНЕТИЧЕСКИЙ ПОИСК**

Белок представляет собой большую молекулу, вырабатываемую в клетке живого организма. Каждый белок состоит из аминокислот, соединенных в цепочку. Существует 20 различных аминокислот со сложной химической структурой, различные комбинации которых создают молекул белков. Для простоты понимания генетики каждой из 20 аминокислот присвоена буква алфавита: Глицин - G, Лейцин - L, Аланин - A, и т.д. Используются следующие 20 букв:



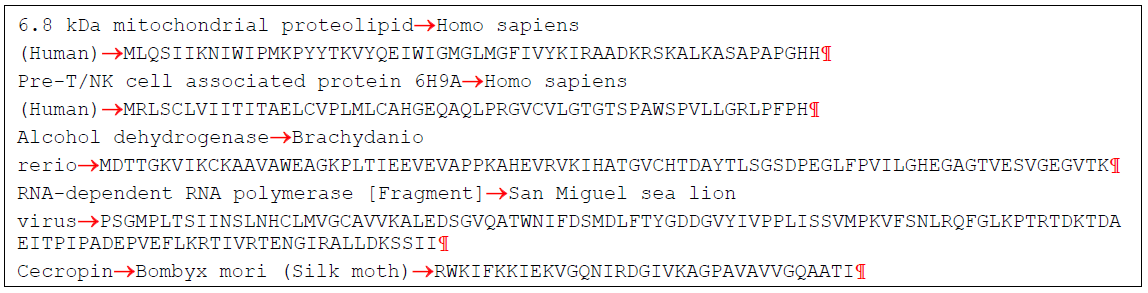
Например, следующая цепочка аминокислот взята из белка “C-FLIP AMPA glutamate”, который содержится в [золотой рыбке](https://ru.wikipedia.org/wiki/%D0%97%D0%BE%D0%BB%D0%BE%D1%82%D0%B0%D1%8F_%D1%80%D1%8B%D0%B1%D0%BA%D0%B0):



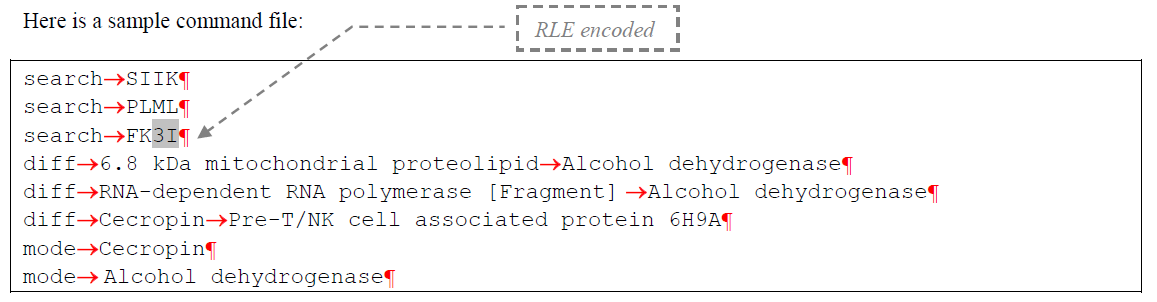
**Формат входных файлов**

В этом проекте используются два входных файла. Один файл содержит генетические данные - sequences.txt, а другой содержит команды, которые необходимо выполнить - commands.txt.

Во входном файле sequences.txt каждая строка содержит название белка, название организма, в котором он обнаружен, и цепочку аминокислот. Данные разделены символом табуляции. На рисунке знаком → обозначен символ табуляции, а знаком ¶ - символ конца строки.



Во входном файле commands.txt каждая строка файла содержит название операции, которую необходимо выполнить над входными данными файла sequences.txt. Возможные операции: search, diff, mode. У каждой операции есть параметры.



Цепочка аминокислот может быть закодирована с использованием простой техники сжатия Run-Length Encoding (RLE) - кодирование повторов. RLE - это алгоритм сжатия данных, заменяющий повторяющиеся символы (серии) на один символ и число его повторов. Серией называется последовательность, состоящая из нескольких одинаковых символов. Рассмотрим следующий фрагмент гена обыкновенной плодовой мухи:

**AAAAAAAATATTTCGCTTTTCAAAAATTGTCAGATGAGAGAAAAAATAAAA**

Кодирование этого фрагмента с использованием алгоритма RLE дает следующую последовательность:

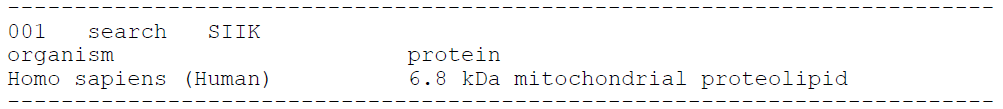
**8ATA3TCGC4TC5ATTGTCAGATGAGAG6AT4A**

Обратите внимание, что последовательностям из 1 или 2 одинаковых букв не предшествует цифра. Такие серии не кодируются, так как это не приведет к сжатию данных. Считайте, что ни одна белковая последовательность не содержит цепочек, состоящих более чем из 9 одинаковых букв.

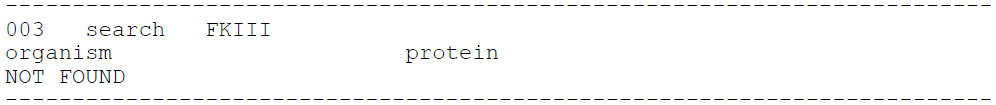
**Операция** **search** имеет следующий формат записи:

**search**<tab><**последовательность аминокислот**><newline>

Выполнение операции **search** означает, что во входных данных из файла sequences.txt необходимо найти указанную <**последовательность аминокислот**>. Для каждого белка, содержащего эту последовательность, необходимо записать в выходной файл genedata.txt название организма и соответствующий белок. Если совпадений не найдено, то записать сообщение “**NOT FOUND**”. В примере ниже, последовательность аминокислот **SIIK** найдена в организме **Homo sapiens (Human)** в белке **6.8 kDa mitochondrial proteolipid**.



А следующий пример вывода означает, что последовательность аминокислот **FKIII** не обнаружена ни в одном белке входного файла:

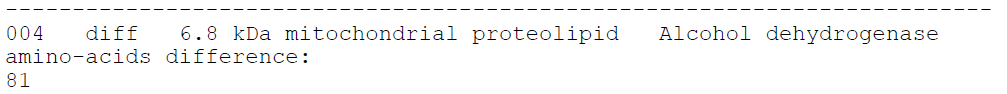


Выводимые в начале строки данные (001 или 003) - это порядковые номера операций по ходу их чтения из файла commands.txt.

**Операция** **diff** имеет следующий формат записи:

**diff**<tab><**белок 1**><tab><**белок 2**><newline>

Выполнение операции **diff** означает, что необходимо определить, сколько аминокислот необходимо заменить, чтобы генетически превратить **белок 1** в **белок 2**. Это можно легко сделать, сравнив аминокислотные цепочки обоих белков и посчитав количество позиций, в которых эти две последовательности различаются. Для белков из примера ниже количество таких позиций равно 81.

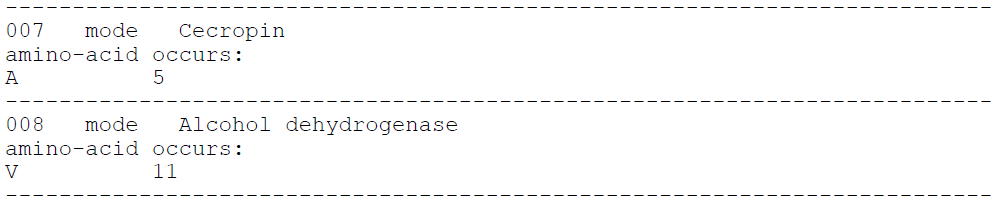


Учитывайте, что длины цепочек **белка 1** и **белка 2** чаще всего различны. Если **белок 1** или **белок 2** не найден в файле исходных данных sequences.txt, то после строки «**amino-acids difference:**» должно быть выведено сообщение «**MISSING:**», за которым следует название(я) отсутствующего(их) белка(ов).

**Операция** **mode** имеет следующий формат записи:

**mode**<tab><**название белка**><newline>

Выполнение операции **mode** означает, что необходимо найти во входных данных указанный белок, а в его цепочке найти аминокислоту, которая встречается чаще всего. Если найдется более одной такой аминокислоты, то выведите ту, которая стоит первой в алфавитном порядке. В примере ниже в белке **Cecropin** чаще всего (5 раз) встречается аминокислота **A**. А в белке **Alcohol dehydrogenase** чаще всего (11 раз) встречается аминокислота **V**.



Если белок не найден в исходном файле, то в строке, следующей за «**amino-acid occurs:**», выведите сообщение «**MISSING:**», за которым следует название отсутствующего белка.

**Формат выходного файла и пример:**

Ваша программа должна выполнить все операции из файла commands.txt над данными из файла sequences.txt и записать результат выполнения этих операций в файл genedata.txt. При записи в выходной файл RLE-кодирование не используется. То есть, если у операции **search** был параметр **FK3I**, то в выходной файл он записывается как **FKIII**.

Первая строка файла должна содержать только ваше имя; вторая строка должна содержать заголовок «Генетический поиск». Следующие строки содержат данные об операциях и результатах их выполнения. Описание операции начинается с трехзначного номера операции, заполненного нулями (001, 002, 003, …, 009, 010, 011,..). Для удобства чтения вывод данных по каждой операции разделяется линиями знаков тире.

Для операции **search**:

• выводим название организма и название белка

• если совпадений не найдено, выводим «NOT FOUND»

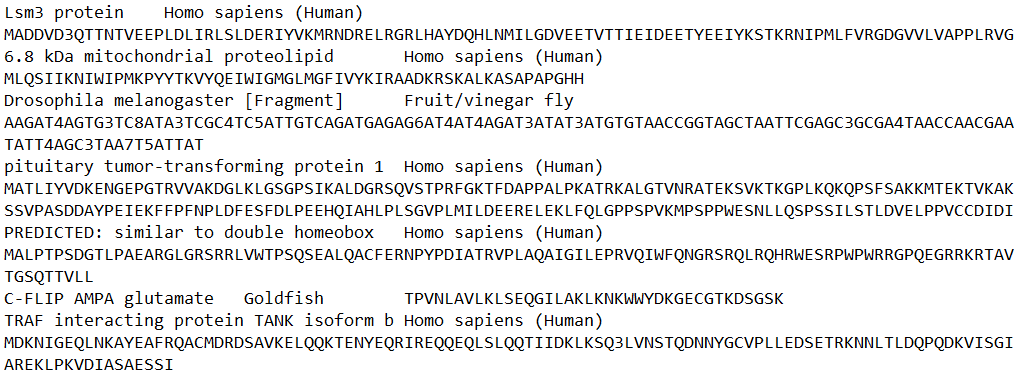
Для операции **diff**:

• выводим количество аминокислот, которыми различаются белки

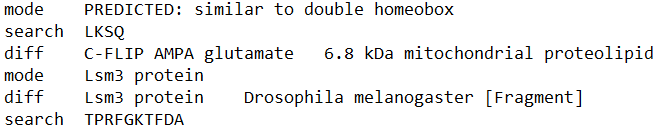
Для операции **mode**:

•выводим наиболее часто встречающуюся аминокислоту и количество раз, которое она встречается в организме.

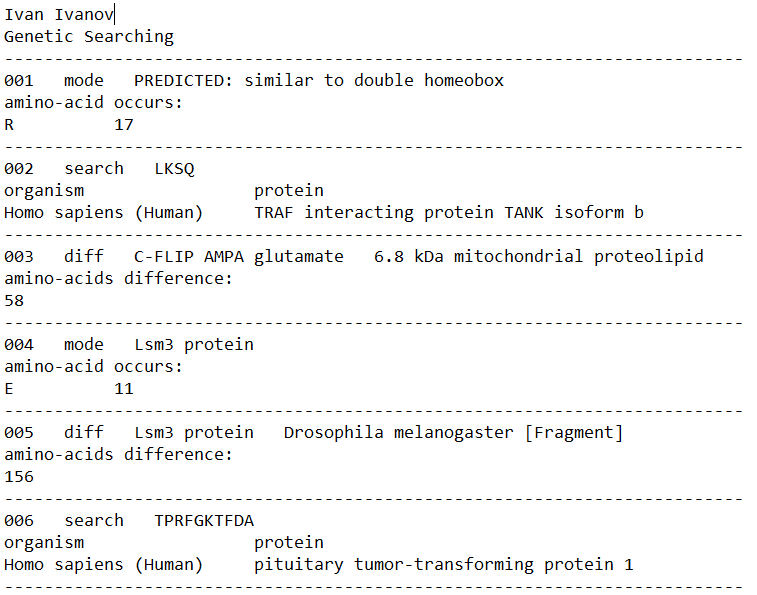
**Пример входного файла sequences.txt**

****

**Пример входного файла commands.txt**



**Пример выходного файла genedata.txt для исходных данных, представленных выше:**



Протестируйте свое решение на предложенных файлах входных и выходных данных: sequences.0.txt, commands.0.txt, genedata.0.txt, sequences.1.txt, commands.1.txt, genedata.1.txt, sequences.2.txt, commands.2.txt, genedata.2.txt

**Замечания по реализации:** Для хранения данных о белках целесообразно использовать коллекцию List<GeneticData>, где GeneticData - структура, позволяющая хранить данные о белках. Структуру необходимо предварительно объявить:

struct GeneticData

{

public string protein; //название белка

public string organism; //название организма

public string amino\_acids; //цепочка аминоскислот

}

Данные об операциях из файла commands.txt сохранять в коллекцию/массив не нужно. Операции выполняются по мере их чтения из файла путем вызова соответствующего метода. Для этого рекомендуется разработать методы для каждой операции.

Не забывайте, что если цепочка аминокислот сжата алгоритмом RTE, то ее необходимо предварительно раскодировать. И наоборот, если есть такая необходимость. Для этого рекомендуется разработать методы:

static string RLEncoding(string amino\_acids)

static string RLDecoding(string amino\_acids)

Так как входные данные представлены строками, то используйте методы класса String для обработки строк (поиск в строке, поиск подстроки и т.д.). Учитывайте, что каждая операция по модификации объекта типа String создает новую строку, предыдущая версия строки остается в памяти и образует так называемый мусор (неиспользуемые объекты). В этом случае эффективнее использовать строки типа StringBuilder.