**Вернуть в виде списка самую длинную подпоследовательность подряд идущих чисел, сумма которых максимальна (задача имеет смысл, если в исходной последовательности будут встречаться, как положительные, так и отрицательные элементы). Если будет несколько таких подпоследовательностей с одинаковой суммой, выбрать самую короткую. Если и длина будет совпадать, то вернуть первую (от начала списка) такую подпоследовательность. Реализовать поиск такой подпоследовательности в один проход (один цикл, второй цикл будет использоваться для копирования элементов найденной**

**подпоследовательности в результирующий список).**

**Реализовать в виде функции**

**public static List<Integer> createNewList(List<Integer> list)**

Входные данные для этой задачи должны читаться из текстового файла, а выходные – записываться в текстовый файл.

В файле каждый набор чисел (массив / список), а также другие параметры, если они предусмотрены условием задачи, должны быть записаны на отдельной строке.

Данная задача должна быть оформлена двумя способами:

1) в виде консольного приложения с разбором параметров командной строки,

2) в виде оконного приложения, где двумерный массив можно задать в JTable. При этом должна быть возможность загрузить данный из файла в JTable (реализованная в виде двух функций: чтение данных из файла в двумерный массив и отображение двумерного массива в JTable), а также сохранить данные из JTable в файл (реализованная в виде двух функций: чтение данных из JTable в двумерный массив и запись двумерного массива в файл).

Функции, реализующие логику задачи и чтение / запись данных из файлов / в файлы, должны быть оформлены в виде отдельного модуля (в отдельном файле). Этот модуль без каких-либо изменений должен использоваться в двух программах: с консольным интерфесом (файлы для чтения / записи задаются в параметрах командной строки) и оконным интерфейсом.

Заранее придумать не менее 10 различных тестов, охватывающих как типичные, так и все возможные граничные (наиболее невероятные и показательные) ситуации. (Сохранить в текстовых файлах input01.txt, input02.txt и т. д.)

Решение, естественно, должно быть оформлено в виде отдельной функции. В реализации обязательно использовать вспомогательные функции.

Запрещено использовать любые стандартные (уже реализованные в библиотеке языка Java) функции и методы массивов и списков (за исключением создания списка, получения размера и добавления нового элемента). Аналоги стандартных функций необходимо реализовать (какие – для каждого варианта указано отдельно).

Реализация в виде консольного приложения с разбором параметров командной строки

Имена файлов для чтения и записи должны передаваться в параметрах командной строки, например, так:

> java ru.vsu.cs.course1.Task8 .\input.txt .\output.txt

(Здесь "." означает текущую директорию, т.е. ".\" можно опустить. Также при обращении к файлам можно использовать "..", что будет означать родительскую директорию относительно текущей, например, "..\..\input05.txt")

Еще лучше будет, если вы реализуете разбор параметров командной строки, чтобы имена входных и выходных файлов можно было задавать в виде именованных параметров, например, так:

> java ru.vsu.cs.course1.Task8 -i .\input.txt -o .\output.txt

или так (а лучше и так и так)

> java ru.vsu.cs.course1.Task8 --input-file=.\input.txt –-output-file=.\output.txt

Если программа запускается без указания необходимых аргументов, либо отсутствует входной файл, то в поток ошибки (System.err.println()) должно печататься сообщение об ошибке и программа должна завершаться с кодом, отличным от 0 (код успешного

завершения).

Для разбора параметров командной строки реализовать функцию:

public static InputArgs parseCmdArgs(String[] args)

InputArgs – класс, в котором описаны поля inputFile и outputFile (и, возможно, какие-от другие параметры в зависимости от задачи).

По возможности перебирать элементы списка циклом for(Integer v: list )