CSC 172: Lab 4
Fall 2018

Released: **09/30/2018**
Due: **10/15, 8AM**

Title: **DNA Strings in Linked List**

A key element in many bioinformatics problems is the biological sequence. A biological sequence is just a list of characters chosen from some alphabet. Two of the common biological sequences are DNA (composed of the four characters A, C, G, and T) and RNA (composed of the four characters A, C, G, and U).

Objective: In this project, you will implement some basic functionality for manipulating DNA and RNA sequences.

Implementation:

You will implement sequences using linked lists, storing one letter of the sequence per linked list node. You may implement either a singly linked list or a doubly linked list, whichever you prefer.

You may implement your linked list using the code from the lecture (available on Blackboard) or you may write your list code from scratch.

In addition to the linked lists of sequences, you will maintain a "sequence array" which stores the various sequences. Commands that manipulate sequences will refer directly to entries in the sequence array. The sequence array will store the sequence **type** (RNA or DNA) and a pointer or other form of access to the linked list that stores the sequence itself.

The **type** field should use an enumerated type variable, and you should also have an enumeration value to recognize that a given position in the sequence array is unused.

All indexing (both for the sequence array and for positions in a sequence) will begin with position *zero*.

The primary design consideration for this project will be the interface between the list class and its client. If you reuse the code from lecture (book), it is acceptable to alter or add some list methods. Regardless of whether you reuse the book code or write your own, it is mandatory that the list class remain *application independent*. No bioinformatics-related code should be part of the list class. All such code should be in some class above the level of the list class.

Input and Output:

The program will be invoked from the command-line as:

```
biol <array-size> <command-file>
```

The name of the program is `biol` (your **DNAList** program goes here). Parameter array-size is the size of the sequence array, and `command-file` is the name of the input file that holds the commands to be processed by the program.

The input for this project will consist of a series of commands (some with associated parameters, separated by spaces), one command for each line. A blank line may appear anywhere in the command file, and any number of spaces may separate parameters. You need not worry about checking for syntactic errors. That is, only the specified commands will appear in the file, and the specified parameters will always appear. However, you <u>must check</u> for logical errors. These include attempts to access out-of-bounds positions in the sequence array or in a sequence. The commands will be read from standard input, and the output from the commands will be written to standard output. The program should terminate after reading the EOF mark. The commands are as follows:

insert *pos type sequence*

Insert sequence to position **pos** in the sequence array. **type** will be either DNA or RNA. You must check that sequence contains only appropriate letters for its type, if not the insert operation is in error and no change should be made to the sequence array. If there is already a sequence at **pos** and if sequence is syntactically correct, then the new sequence replaces the old one at that position.

It is acceptable that sequence be *null* (contain no characters) in which case a null sequence will be stored at **pos**. Note that a *null* sequence in a sequence array slot is different from an empty slot.

remove *pos*

Remove the sequence at position **pos** in the sequence array. Be sure to set the **type** field to indicate that this position is now empty. If there is no sequence at **pos**, output a suitable message.

print

Print out all sequences in the sequence array. Indicate for each sequence its position within the sequence array and the type of that sequence (RNA or DNA). Don't print anything for slots in the sequence array that are empty.

print *pos*

Print the sequence and type at position **pos** in the sequence array. If there is no sequence in that position, print a suitable message.

clip *pos start end*

Replace the sequence at position **pos** with a clipped version of the sequence. The clipped version is that part of the sequence beginning at character **start** and ending with character **end**. It is an error if **start** has a value less than zero, or if **start** or **end** are beyond the end of the sequence. A clip command with such an error should make no alteration to the sequence. If

there is no sequence at this slot, output a suitable message. If the value for **end** is less than the value for **start** then the result should be a sequence containing no characters.

copy *pos1 pos2*

Copy the sequence in position **pos1** to **pos2**. If there is no sequence at **pos1**, output a suitable message and do not modify the sequence at **pos2**.

transcribe *pos1*

Transcription converts a DNA sequence in **pos1** to an RNA sequence. It is an error to perform the transcribe operation on an RNA sequence. To transcribe a DNA sequence, change its type field to RNA, convert any occurrences of T to U, complement all the letters in the sequence, and reverse the sequence. Letters A and U are complements of each other, and letters C and G are complements of each other. If the slot is empty, then print a suitable message.

Submission

Submit source code from this lab at the appropriate location for **Lab 4** on the Blackboard system at *learn.rochester.edu*. You should hand in a single **zip** (compressed archive) **.zip** containing your source code and README files, as described below. Do not forget to include **FirstName** and **LastName** (+partner) in the filename. Your **main** (executable) class must be named **DNAList.java**

The file named **README** must include your contact information, your partner's name (if any), a brief explanation of the lab (a one paragraph synopsis. Include information identifying what class and lab number your files represent.), and one sentence explaining the contents of any other files you hand in.