# Programming Project #1

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

**Implementation:**

You will implement sequences using any of the python data types.

In addition, your program will maintain a "sequence array" which stores the various DNA/RNA sequences. The array size is **100 (0 ~ 99)**. 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 RNA/DNA sequence itself. The type field should also be able to indicate 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.

**Input and Output:**

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

```
python3 bio.py <command-file>
or
python3 bio.py <array size> <command-file>
```

The name of the program is **bio.py**. 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.

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 must check 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 safe to assume that sequence is **NOT** null (containing no characters).

**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.

**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.

**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.

**swap** pos1 start1 pos2 start2

Swap the tails of the sequences at positions *pos1* and *pos2*. The tail for *pos1* begins at character *start1* and the tail for *pos2* begins at character *start2*. It is an error if the value of the start position is greater than the length of the sequence or less than zero. If the length of a sequence is n, the start position may be n, meaning that the tail of the other sequence is appended (i.e., a tail of null length is being swapped). The swap operation should be reported as an error if the two sequences are not of the same type, or if one of the slots does not contain a sequence. In either case, no change should be made to the sequences.

**transcribe** *pos1*

Transcription converts a DNA sequence 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.

**Sample Test Data:**

```
insert 2 DNA AGG
insert 3 DNA AGGC
insert 5 RNA ACCU
print
remove 1
remove 2
print
insert 16 DNA AGCGGCG
insert 15 RNA AAA
print  2
print 15
copy 15 2
copy 17 15
print 2
print 15
swap 16 3 15 4
print 15
print 16
transcribe 5
print
```

**CHEATING**: Cheating of any kind will **NOT** be tolerated. Any thing that is handed in must be your own work! However, talking with one another to understand the material better is strongly encouraged. Recognizing the distinction between cheating and cooperation is very important.

- Googling for a solution is allowed
- If someone helps you understand the problem, you are not cheating. We strongly encourage students to help one another understand the material presented in class, in the book, and general issues relevant to the assignments. Just do **not copy** the solutions.
- If you **copy** someone else's solution, you are cheating. If you let someone else copy your solution, you are cheating. We will not distinguish between the person who copied a solution and the person whose solution was copied. Both people will be treated as cheaters.