Programming Assignment #3

Background Information

Deoxyribonucleic acid (DNA) is a nucleic acid containing the genetic instructions used in the development and functioning of all known living organisms. The DNA segments carrying this genetic information are called genes. Likewise, other DNA sequences have structural purposes, or are involved in regulating the use of this genetic information. Along with RNA and proteins, DNA is one of the three major macromolecules that are essential for all known forms of life.

A DNA string, also called DNA strand, is a finite sequence consisting of four letters, A, C, G, and T in any order. The four letters stand for the four nucleotides: adenine, cytosine, guanine, and thymine. Nucleotides, which are the molecular units from which DNA is composed, are also called bases.

Genes, which are made up of DNA, are the basis for heredity in living organisms and act as instructions to make molecules called proteins. Proteins are made from DNA through the processes of transcription (DNA to mRNA) and translation (mRNA to Protein chain).

Bioinformatics plays an important role in identifying genes within a long DNA sequence. Instead of manually studying the behavior of genes in an organism or in a lab, which can be very time consuming, bioinformatics and computers can provide educated guesses about where genes are located (known as gene prediction) by analyzing DNA sequence data in a fraction of the time.

Gene prediction, as described in the "Bioinformatics: Finding Genes" link below, is a complex, multi-faceted process. But, one of the key indicators we can look for are called open reading frames: stretches of DNA that, when read the nucleotides are read in triplets (a codon), contain no termination codons and so can translate as a polypeptide chain.

See the following links for more information:

- http://en.wikipedia.org/wiki/DNA
- http://en.wikipedia.org/wiki/Open_reading_frame
- http://en.wikipedia.org/wiki/Reading_frame
- Bioinformatics: Finding Genes: http://www.genome.gov/25020001
- Interactive Transcription and Translation: http://learn.genetics.utah.edu/content/molecules/transcribe/
- Transcription and Translation Tool: http://www.attotron.com/cybertory/analysis/trans.htm
- Online ORF Finder: http://www.bioinformatics.org/sms2/orf_find.html

Program Description

You will write a program that searches for open reading frames (ORFs) in a supplied DNA sequence. The program will output the results of the ORF search, including translation of the any ORFs to amino acid sequences. The "Online ORF Finder" link above is a similar online program that you might try out to make sure you understand what should be done.

The program will open an input file (the filename provided as a command line argument) that contains the source DNA sequence. If the program successfully opens the file, it will parse the sequence description and data out of it.

The source sequence will need to be searched for ORFs over the three possible reading frames of both the direct strand (provided) and the reverse/compliment strand (which must be determined). Each of the ORFs will also have a polypeptide chain, found by transcription and translation of its DNA segment. Once all the ORFs have been found, the program will produce a report, saved to an output file (the filename provided as additional command line argument).

Computational Tasks

Your program must perform the following tasks:

1. Determine the reverse/complimentary strand of a DNA sequence
2. Determine the open reading frames for the three possible reading frames on the direct strand and also the three possible reading frames on the reverse/compliment strand
3. Determine the translated protein sequence for each of the open reading frames found

Determine reverse/complimentary strand of DNA

The direct strand of DNA will be given in the 5'->3' direction. The reverse/complimentary strand is aligned with the direct strand in the opposite direction (3'->5') and aligned base pairs formed are complimentary bases to each other (A ⇔ T, C ⇔ G). For example, with the following direct strand:

```
5' - TCAATGTAACGCGCTACCCGGAGCTCTGGGCCCAAATTTCATCCACT - 3'
```

The reverse strand would be:

```
3' - AGTTACATTGCGCGATGGGCCTCGAGACCCGGGTTTAAAGTAGGTGA - 5'
```

Determine open reading frames

An open reading frame is a contiguous sequence of nucleotide triplets (codons) that (a) starts with the "start codon" ATG, (b) ends with a "stop codon" (TAA, TAG, or TGA), (c) does not contain any other stop codon that is "in-frame" (respecting the triplet boundaries), and (d) is as long as possible subject to the conditions (a)-(c).

The search for open reading frames happens by reading a strand in the 5'->3' direction and there are three possible reading frames (per strand). In the above example, the direct strand would be read from left to right with reading frame 1 starting on T, reading frame 2 starting on C and reading frame 3 starting on A. The reverse strand would be read from right to left with reading frame 1 starting on A, reading frame 2 starting on G and reading frame 3 starting on T.

There is an open reading frame in that example, found on the reverse strand in reading frame 3. The start codon (ATG) is colored green and the stop codon (TGA) is colored red (remember you read it from 5'->3'):

```
3' - AGTTACATTGCGCGATGGGCCTCGAGACCCGGGTTTAAAGTAGGTGA - 5'
```

To confirm your work, I recommend you try the Online ORF Finder mentioned at the end of the Background section above.

Determine the translated protein sequence for each ORF

To translate the DNA segment of an ORF into its protein components, it is first necessary to transcribe the DNA to mRNA. The ORF's DNA segment would be considered the "coding strand" and the its reverse/complimentary segment considered the "template strand," because RNA for the "coding strand" is actually generated as the

complement of the "template strand". RNA is similar to DNA, except RNA includes the nucleotide uracil (U) where thymine (T) would have been used in the DNA. A short cut to determining the mRNA sequence for any DNA sequence, which bypasses the "coding" and "template" strand notions, would be to copy the DNA sequence, then replace all of the thymine with uracil.

Given the previous example of an open reading frame:

```
5' - ATG AAA TTT GGG CCC AGA GCT CCG GGT AGC GCG TTA CAT TGA - 3'
```

The mRNA would be:

```
5' - AUG AAA UUU GGG CCC AGA GCU CCG GGU AGC GCG UUA CAU UGA - 3'
```

Now, we can translate the codons in the mRNA to the proteins they encode for. Using the following RNA codon to Protein mapping table, we can find the final amino acid sequence. Single letter abbreviations for the encoded proteins (with an asterisk indicating "stop") are used in the final sequence (a list of the abbreviations can be found here: http://www.biochem.ucl.ac.uk/bsm/dbbrowser/c32/aacode.html).

First Base	Second Base: U	Second Base: C	Second Base: A	Second Base: G	Third Base
U	UUU Phenylalanine (Phe)	UCU Serine (Ser)	UAU Tyrosine (Tyr)	UGU Cysteine (Cys)	U
	UUC Phenylalanine (Phe)	UCC Serine (Ser)	UAC Tyrosine (Tyr)	UGC Cysteine (Cys)	C
	UUA Leucine (Leu)	UCA Serine (Ser)	UAA Stop	UGA Stop	A
	UUG Leucine (Leu)	UCG Serine (Ser)	UAG Stop	UGG Tryptophan (Trp)	G
C	CUU Leucine (Leu)	CCU Proline (Pro)	CAU Histidine (His)	CGU Arginine (Arg)	U
	CUC Leucine (Leu)	CCC Proline (Pro)	CAC Histidine (His)	CGC Arginine (Arg)	C
	CUA Leucine (Leu)	CCA Proline (Pro)	CAA Glutamine (Gln)	CGA Arginine (Arg)	A
	CUG Leucine (Leu)	CCG Proline (Pro)	CAG Glutamine (Gln)	CGG Arginine (Arg)	G
A	AUU Isoleucine (Ile)	ACU Threonine (Thr)	AAU Asparagine (Asn)	AGU Serine (Ser)	U
	AUC Isoleucine (Ile)	ACC Threonine (Thr)	AAC Asparagine (Asn)	AGC Serine (Ser)	C
	AUA Isoleucine (Ile)	ACA Threonine (Thr)	AAA Lysine (Lys)	AGA Arginine (Arg)	A
	AUG Methionine (Met) or Start	ACG Threonine (Thr)	AAG Lysine (Lys)	AGG Arginine (Arg)	G
G	GUU Valine (Val)	GCU Alanine (Ala)	GAU Aspartic acid (Asp)	GGU Glycine (Gly)	U
	GUC Valine (Val)	GCC Alanine (Ala)	GAC Aspartic acid (Asp)	GGC Glycine (Gly)	C
	GUA Valine (Val)	GCA Alanine (Ala)	GAA Glutamic acid (Glu)	GGA Glycine (Gly)	A
	GUG Valine (Val)	GCG Alanine (Ala)	GAG Glutamic acid (Glu)	GGG Glycine (Gly)	G

The table can be found in the website for Biological and Environmental Research Information System, Oak Ridge National Laboratory:
https://public.ornl.gov/site/gallery/detail.cfm?id=460&topic=&citation=32&general=&restsection=public.

The resulting protein sequence for the example open reading frame is then:

```
M  K  F  G  P  R  A  P  G  S  A  L  H  *
```

To confirm your work, I recommend you try the Online ORF Finder mentioned at the end of the Background section above.

While lengthy, the most straightforward approach to "translation" of the RNA codon to an amino acid would be to use nested switch statements, where you branch according to the matrix above.

Command Line

When run, the program must be provided with the name of an input file and the name of an output file:

```
PROGRAM_NAME <INPUT_FILE> <OUTPUT_FILE>
```

The angle brackets indicate a mandatory argument. The brackets themselves should not be included on the command line.

If not provided with the required number of arguments, the program should display an appropriate error message on the standard error stream and exit.

Input

There is one input file for the program; the name provided on the command line. The program will need to open and read the data from the file. If the file does not exist or cannot be opened for any reason, the program should display an error message on the standard error stream and exit. The input file should have the format described below; if it has incorrectly formatted data, the program should display an appropriate error message on the standard error stream and exit.

DNA sequence input file

The DNA sequence input file will contain a single direct strand (5'->3') DNA sequence in FASTA format. This format requires a single-line description followed by one or more lines of sequence data. The description line will begin with a ">" character, followed immediately by the description of the sequence (no space in between). The data of the sequence will begin on the subsequent line and will likely be split onto multiple lines.

- There may be any number of blank lines before the description line and the description line must be the first non-blank line in the file
- The total sequence data must be at least 15 characters long and will not be longer that 5000 characters
- A line of sequence data may be split into smaller segments with whitespace between
- A line of sequence data will be less than 100 characters long
- The characters used in the sequence may only be A, C, T and G and may be in upper or lower case
- The sequence provided should be considered the direct strand; the beginning of the sequence is the 5' end and the end of the sequence is the 3' end

A sample input file might look like the following:

```
>sample sequence xyz
TTGTACCTGTATATGTATTAATACTATATTGCTGCAGATAAAATTTTATAATTCAAAATT
AGAGATCAACGAAATAATTGATATGCCTTATGCAACATTTTTAGTTCTTTGATGTAATTC
ATCTTAACAGGTACCATTATGTATATGCAGCATATAATCAAAA
```

Another valid input file might look like:

```
>segment of Drosophila genome
ccacctcgga gtcaacctgc tacggtcata ccgcacgcac ttcagtggta atcacatccc
aatccgcagc aaaacaaaga ataaccatga accgctacgc ggtaagctcg atggtggggc
aaggatcctt cgggtgcgta tacaaggcga cacgcaagga cgacagcaag gtggtggcca
tcaaagtgat ctccaaggtg agtggggcgg gccaggtgat aaagcaacaa gtccatacaa
```

Output

The program will produce a single output file; the name provided on the command line. If the file already exists, the program should overwrite its contents. If the file cannot be opened for any reason, the program should display an error message on the standard error stream and exit. The format of your output file is described below.

DNA analysis output file

The output file will contain the ORF search results. There will be an initial header and then the ORF results will be written to the output file in sections, with one section for every reading frame on the direct strand, followed by the reverse strand. If no ORFs were found for a reading frame, a single line will be output to that effect. Otherwise, for each ORF, the details about the ORF will be output, as well as the protein translation of the ORF.

The header for the results must have this format:

```
ORF results for "SEQUENCE_DESCRIPTION" containing MMM bases
```

- SEQUENCE_DESCRIPTION should match the original description from the input file and should be preceded by a single ">" character
- MMM is the count of the number of nucleotide bases from the source DNA sequence

If a reading frame did not contain any ORFs, the following should be output for its section:

```
No ORFs were found in reading frame N on the STRAND strand
```

- N is the number of the corresponding reading frame (1, 2 or 3)
- STRAND will be either "direct" or "reverse"

For each ORF that was found, the following should be output:

```
>ORF number M in reading frame N on the STRAND strand extends from base S to base E
ORF_SEQUENCE

>Translation of ORF number M in reading frame N on the STRAND strand
ORF_SEQUENCE_PROTEIN_TRANSLATION
```

- The details line for the ORF should begin with a single ">" character
- M is the number of the ORF on the reading frame, indicating the order it was found in (when reading in the 5'->3' direction). The first ORF found (in each reading frame) will be numbered 1
- N is the number of the corresponding reading frame (1, 2 or 3)
- STRAND will be either "direct" or "reverse"
- S and E correspond to the position of the starting and ending bases for the ORF in the source DNA sequence (1 being the first position)
- ORF_SEQUENCE is the raw sequence data of the ORF using upper-case A, C, G and T characters. A line of the sequence should not be longer than 79 characters, so may need to be output on multiple lines
- The details line for the translation of the ORF should begin with a single ">" character
- ORF_SEQUENCE_PROTEIN_TRANSLATION is the sequence of amino acids that the ORF translates to, using the single letter abbreviations of each protein (with an asterisk for the stop codon). A line of the sequence should not be longer than 79 characters, so may need to be output on multiple lines

If given the following sequence in the input file:

```
>sample sequence abc
TCAATGTAACGCGCTACCCGGAGCTCTGGGCCCAAATTTCATCCACT
```

The output file would look like the following:

```
ORF results for "sample sequence abc" containing 47 bases

No ORFs were found in reading frame 1 on the direct strand

No ORFs were found in reading frame 2 on the direct strand

No ORFs were found in reading frame 3 on the direct strand

No ORFs were found in reading frame 1 on the reverse strand

No ORFs were found in reading frame 2 on the reverse strand

>ORF number 1 in reading frame 3 on the reverse strand extends from base 6 to base 47
ATGAAATTTGGGCCCAGAGCTCCGGGTAGCGCGTTACATTGA

>Translation of ORF number 1 in reading frame 3 on the reverse strand
MKFGPRAPGSALH*
```

User Interface

There is no user interface for this program. The only interaction is specifying the input and output filenames as arguments on the command line.

Grading

The program will be graded based on the following rubric:

- Does the program compile (on one of the cslab machines): 15%
- Correctness and implementation of the DNA sequence class: 10%
- Correctness and implementation of the open reading frame class: 10%
- Correctness and implementation of the ORF search task: 10%
- Correctness and implementation of the transcription/translation task: 10%
- Correctness and implementation of the file I/O operations: 15%
- Correctness and implementation of the main function and any other functions in the program: 20%
- Correctness and use of multiple file implementation: 10%
- Documentation: 10%
- Style (variable naming, indentation, etc…): 5%

Programming Rules

Your program must adhere to the programming rules as described in the "Programming Rules and Guidelines" document posted on Blackboard. Of note, or in addition:

Your program must be well documented. This includes having a preamble, prologue for every function prototype (describing parameters, pre- and post-conditions, and return value), and comments throughout the code (describing any non-trivial algorithms in plain English).

Your program must follow common stylistic guidelines that cover line spacing, whitespace, indentation, and choosing appropriate variable/function/class names using consistent naming conventions. You may not use global variables, unless they are constants.

You must implement and use a class to represent a DNA sequence (one object should contain both direct and reverse/complimentary strands).

You must implement and use a class to represent a *single* open reading frame (ORF).

You'll need to determine the member variables and member functions your classes will need and which members will be made public versus private. As you make those decisions, remember that your class must respect the OOP principles for access, information hiding, data abstraction and encapsulation.

All input and output stream objects must be created in your main() function. Opening, closing and error handling for all input and output files must be performed in your main() function, as well. All other I/O operations *should* be performed in your main() function (or functions specifically designed for it, that do not perform any other tasks).

The program must read the names of input and output files from the command line (you cannot hard-code the names of the files into the program). You are expected to handle a missing input file gracefully and verify the input data as valid, as described in the Input section above. Any problems encountered attempting to open the output file should also be handled gracefully, as described in the Output section above.

All output should be sent to a corresponding file output stream, except for error messages. Any error messages you display should be sent to the standard error stream. A successful execution of your program should have no console output.

You must use multiple source code files for your program (separate compilation). Each class you create must have its own header and implementation files. The source code file containing your main function must be called **`main.cpp`**.

What to submit and when

This assignment is due by the end of the day (i.e. 11:59PM, EST) on Friday, December 12th, 2014.

Before you submit the assignment, make sure that it compiles and runs correctly on one of the cslab machines. Do not enhance your program beyond this specification. Do not make it do anything except what is described in this document.

You need to submit multiple files for this assignment, so they must be placed into a directory and zipped up as instructed below. The name of this directory must be **`lastname_firstname_hwk3`** (replace **`lastname`** and **`firstname`** with your last and first name, *using lowercase letters*). Do not name it anything else. If it is named anything else, you lose 3% of the program's value.

You are to create the above named directory and place all of your source code files into it. The source code file that contains your main function must be called **`main.cpp`** (you will lose 1% if it is not). Do not place any executable files, object files, input or output files in this directory. You will lose 1% for each file that does not belong there. With all necessary files in your directory, if you're on a Linux or Mac computer, run the following command from the parent directory that contains your newly created directory:

```
zip -r lastname_firstname_hwk3.zip lastname_firstname_hwk3/
```

This will compress all of your files into a file named **`lastname_firstname_hwk3.zip`**. If you're on a Windows computer, there should be an option to "compress" the folder to a zip archive.

Submit the zip file to the assignment in our Blackboard course. Do not paste the code into your submission.