Fall 2013 CSCI 135. SEC 4 Analysis and Design I. Project 2

Problem Statement (Project Requirement)

For this project, you will refer to the tables (source: http://en.wikipedia.org/wiki/DNA_codon_table) shown belows:

Standard genetic code

1st base	2nd base								3rd base
	T		C		A		G		
T	TTT	(Phe/F) Phenylalanine	TCT	(Ser/S) Serine	TAT	(Tyr/Y) Tyrosine	TGT	(Cys/C) Cysteine	T
	TTC		TCC		TAC		TGC		C
	TTA	(Leu/L) Leucine	TCA		TAA	Stop (*Ochre*)	TGA	Stop (*Opal*)	A
	TTG		TCG		TAG	Stop (*Amber*)	TGG	(Trp/W) Tryptophan	G
C	CTT		CCT	(Pro/P) Proline	CAT	(His/H) Histidine	CGT	(Arg/R) Arginine	T
	CTC		CCC		CAC		CGC		C
	CTA		CCA		CAA	(Gln/Q) Glutamine	CGA		A
	CTG		CCG		CAG		CGG		G
A	ATT	(Ile/I) Isoleucine	ACT	(Thr/T) Threonine	AAT	(Asn/N) Asparagine	AGT	(Ser/S) Serine	T
	ATC		ACC		AAC		AGC		C
	ATA		ACA		AAA	(Lys/K) Lysine	AGA	(Arg/R) Arginine	A
	ATG[A]	(Met/M) Methionine	ACG		AAG		AGG		G
G	GTT	(Val/V) Valine	GCT	(Ala/A) Alanine	GAT	(Asp/D) Aspartic acid	GGT	(Gly/G) Glycine	T
	GTC		GCC		GAC		GGC		C
	GTA		GCA		GAA	(Glu/E) Glutamic acid	GGA		A
	GTG		GCG		GAG		GGG		G

The first table is very similar to the table you were given in Project 1. There are two exceptions: first, Uracil (U) is replaced with Thymine (T), and second, the translated amino acid is also represented by a single letter abbreviation (for e.g., Q stands for the amino acid Glutamine).

The second table is just an inverse mapping from amino acids to the possible codons:

Inverse table (compressed using IUPAC notation)

Amino acid	Codons	Compressed	Amino acid	Codons	Compressed
Ala/A	GCT, GCC, GCA, GCG	GCN	Leu/L	TTA, TTG, CTT, CTC, CTA, CTG	YTR, CTN
Arg/R	CGT, CGC, CGA, CGG, AGA, AGG	CGN, MGR	Lys/K	AAA, AAG	AAR
Asn/N	AAT, AAC	AAY	Met/M	ATG	
Asp/D	GAT, GAC	GAY	Phe/F	TTT, TTC	TTY
Cys/C	TGT, TGC	TGY	Pro/P	CCT, CCC, CCA, CCG	CCN
Gln/Q	CAA, CAG	CAR	Ser/S	TCT, TCC, TCA, TCG, AGT, AGC	TCN, AGY
Glu/E	GAA, GAG	GAR	Thr/T	ACT, ACC, ACA, ACG	ACN
Gly/G	GGT, GGC, GGA, GGG	GGN	Trp/W	TGG	
His/H	CAT, CAC	CAY	Tyr/Y	TAT, TAC	TAY
Ile/I	ATT, ATC, ATA	ATH	Val/V	GTT, GTC, GTA, GTG	GTN
START	ATG		STOP	TAA, TGA, TAG	TAR, TRA

You will design the class DNASequence with the following interface:

```
class DNASequence {

private:

      // your member variables
      // any other private member functions

public:

      int getLength();
      void clear();

      void appendCodon(char n1, char n2, char n3);
      void appendAminoAcid(char aminoAcid);
      void appendAminoAcidsFromFile(string aFile);

      string getNucleotideSequence();
      string getAminoAcidSequence();

};
```

Here are the detailed specifications:

(a) The objective is to design a `DNASequence` class to represent DNA sequences. The class must provide member functions to append nucleotides in 3 different ways: (1) in the form of codons, (2) in the form of nucleotides from codons that are "translated" from amino acids, and (3) in the form of nucleotides from codons "translated" from amino acids that appear in data files. Finally, the class must also provide member functions to get back the sequence in two forms: (1) as chains of nucleotides, and (2) as chains of translated amino-acids

(b) The `getLength()` member function returns the length of the sequence (number of nucleotides).

(c) The `clear()` member function must clear any internal representation of the sequence and reset appropriate member variables such that, afterwards, if `getLength()` is called, it returns 0.

```
DNASequence seq;
seq.clear();
cout << seq.getLength() << endl; // prints 0
```

(d) The `getNucleotideSequence()` member function returns a string holding the complete nucleotide sequence. If the sequence length is 0, the function must return an empty string **""**. The string returned must be in all upper-case.

```
seq.clear();
cout << seq.getNucleotideSequence() << endl; // prints "" (nothing)
```

(e) The member function `appendCodon()` should take 3 characters representing nucleotides and append it to the internal sequence. The only allowed characters are G, A, T, and C (both upper- and lower- case).

```
seq.appendCodon('G','A','T');
cout << seq.getNucleotideSequence() << endl; // prints GAT
seq.appendCodon('T','T','X'); // nothing is appended, X is invalid
cout << seq.getNucleotideSequence() << endl; // still prints GAT
```

(f) The member function `appendAminoAcid()` takes a character (both upper- and lower- case) representing an amino acid, and appends 3 nucleotides that can translate to this amino acid into the internal sequence. If there are multiple codon sequences that translate to the given amino acid, an arbitrary translation can be picked. If the given amino acid is not recognized, nothing is appended.

```
seq.appendAminoAcid('Q'); // this is Glutamine: CAA can be added
cout << seq.getNucleotideSequence() << endl; // prints GATCAA

seq.appendAminoAcid('Z'); // Z is invalid; nothing appended
cout << seq.getNucleotideSequence() << endl; // still prints GATCAA
```

(g) The member function `appendAminoAcidsFromFile()` will read the contents of the provided file and for each amino acid letter (both upper- and lower- case) found in the file, it will append 3 nucleotides that can translate to it into the internal sequence. If a letter is not recognized, nothing is appended.

In the following example, amino1.txt contains the following: WAZY
[W=TGG, A=GCT (or any of the other 3), Z=invalid, Y=TAT (or TAC)]

```
seq.appendAminoAcidsFromFile("amino1.txt");
cout << seq.getLength() << endl; // prints 15
cout << seq.getNucleotideSequence() << endl;
// prints: GATCAATGGGCTTAT
```

(h) The `getAminoAcidSequence()` member function returns a string that is a translated version of the internally held sequence (in other words, what is returned by `getNucleotideSequence()`). For each codon, the single letter amino acid translation must be used. You must use an all upper-case form when returning the string. You must use * for the START amino acid and ! for the STOP amino acids. Refer to Table 2.

```
seq.clear();
seq.appendCodon('A','T','G');
seq. appendAminoAcidsFromFile("amino1.txt"); // contains WAZY
seq.appendCodon('T','A','A);
cout << seq.getLength() << endl; // prints out 15
cout << seq.getNucleotideSequence() << endl;
// prints out: ATGTGGGCTTATTAA
cout << seq.getAminoAcidSequence() << endl;
//prints out: *WAY!
```

I am placing no particular limits on the length of the sequence in a DNASequence object. I recommend using a C++ string to represent the sequence internally in the class. For the member functions in (f) and (g), you are free to pick any of the codon translations, in case there are multiple mappings (e.g., for S you may pick TCT). For (h), however, your translations must be accurate (e.g., TAC and TAT both translate to Y)

Grading

The points will be distributed as follows:

Criteria	**Maximum Points**
Class design	3
Code comments	3
b-h	21
Test program	3
Total	30

Class Design: Proper encapsulation, separation of interface and implementation, and physical separation of compilation units including proper safeguards.

Code comments: Introductory/title comment, pre- and post- condition comments, other relevant comments.

b-h: implemented according to specifications

Test program: A program with the main() function that demonstrates the use of this class and all its member functions. Test data files containing amino acid sequences will be provided in Blackboard.

You must test and make sure that your program compiles and executes in the Hunter CS Lab environment.

Submission

You will have 2 weeks to work on this project. It is due on Nov 27th (Wed), 2013.

Your deliverables should include multiple files. You can zip the files and submit them as follows.

Suppose you have the following files: **DNASequence.cpp, DNASequence.h, main.cpp, amino.txt**, and your userid is sc, then zip them with this command:

```
zip  sc_p2.zip  DNASequence.cpp  DNASequence.h  main.cpp  amino.txt
```

This will create the zip file sc_p2.zip. Then submit it as follows:

```
cp  sc_p2.zip  /data/biocs/b/student.accounts/sadatc/submissions/cs135/project2
```

Finally, remember to change permissions:

```
chmod 644  /data/biocs/b/student.accounts/sadatc/submissions/cs135/project2/sc_p2.zip
```