**DNA to Protein**

**Cassey Chan, Jennifer Yuzon, Bravolly Pich**

**July 10, 2014**

**Problem Description:**

Many researchers in the field of biology find it increasingly useful to use computer programming to analyze and process data. For example, a molecular biologist may want to identify the protein associated with the gene (DNA sequence) they have isolated in the lab.  This is because DNA itself is just a series of nucleobases, adenine (A), thymine (T), guanine (G) and cytosine (C). In order for the DNA code to have a physiological impact on the organism, the code must be translated into functional proteins that catalyze reactions to support life.  The DNA code is translated by reading every three nucleotides called a codon (i.e. ATG) in the DNA sequence and translate into an amino acid (ATG = M for Methionine)

For example the DNA sequence:

3' end TACCACGTGGACTGAGGACTCCTCTTCAGACGCCAATGACGGGACACCCCGTTCCACTTGCACCTACTTCAACCACCACTCCGGGACCCGTCCGACGACCACCAGATGGGAACCTGGGTCTCCAAGAAACTCAGGAAACCCCTAGACAGGTGAGGACTACGTCAATACCCGTTGGGATTCCACTTCCGAGTACCGTTCTTTCACGAGCCACGGAAATCACTACCGGACCGAGTGGACCTGTTGGAGTTCCCGTGGAAACGGTGTGACTCACTCGACGTGACACTGTTCGACGTGCACCTAGGACTCTTGAAGTCCGAGGACCCGTTGCACGACCAGACACACGACCGGGTAGTGAAACCGTTTCTTAAGTGGGGTGGTCACGTCCGACGGATAGTCTTTCACCACCGACCACACCGATTACGGGACCGGGTGTTCATAGTGATT

5' end

Translates into the amino acid sequence:

MTGHPVPLAPTSTTTPGPVRRPPDGNLGLQETQETPRQVRTTSIPVGIPLPSTVLSRATEITTGPSGPVGVPVETV

This is the beta subunit of hemoglobin. Hemoglobin is a protein in red blood cells that transports oxygen throughout the body.

The purpose of this project is to allow the user to input a DNA sequence and return an amino acid sequence (a potential protein) using either the direct input DNA, its complementary, or reverse sequence. The user can then use the output for DNA functionality studies, DNA or amino acid alignment for phylogenetic analyses and so forth.

**Code Description:**

The DNA class holds and stores the data members for DNA strings and translations.

static public class DNA {

public String dnaStr;

public String dnaRvs;

public String compli;

public String codonStr = "";

public String aminoAcidSqn = "";

public String rvsAminoSqn = "";

public String compliAminoSqn = "";

}

The reverseDNA() function takes in a DNA string and stores each character in an array starting from the end of the string back to the beginning of the string, then converts the array into a string and returns it.

static public String reverseDNA(String myDNA) {

int size = myDNA.length();

char[] dnaAry = new char[size];

String dnaRvs;

int j = (size - 1);

for(int i = 0; i < size; i++) {

dnaAry[i] = myDNA.charAt(j);

j--;

}

dnaRvs = new String(dnaAry);

return dnaRvs;

}

The compliDNA() function takes in a DNA string and uses a loop to replace each character (A -> T, T -> A, G -> C, C -> G) and returns that string.

static public String compliDNA(String myDNA) {

int size = myDNA.length();

char[] dnaAry = new char[size];

String compli;

for(int i = 0; i < size; i++) {

if(myDNA.charAt(i) == 'A') {

dnaAry[i] = 'T';

} else if(myDNA.charAt(i) == 'T') {

dnaAry[i] = 'A';

} else if(myDNA.charAt(i) == 'G') {

dnaAry[i] = 'C';

} else if(myDNA.charAt(i) == 'C') {

dnaAry[i] = 'G';

}

}

compli = new String(dnaAry);

return compli;

}

The createCodonString() takes in a DNA string and returns a codon string. It uses a nested loop where it starts out taking the first 3 letters and store it into the string variable “codon” and checks if that variable matches “ATG”. If the variable matches it goes into the next loop where it continue to store the string by threes until the variable matches either “TAA”, “TGA” or “TAG” where then it will return the stored string and exit.

static public String createCodonString(String myDNA) {

int size = myDNA.length();

String codon = "";

String codonStr = "";

for(int i = 0; i < size - 2; i++) {

codon = String.valueOf(myDNA.charAt(i))

+ String.valueOf(myDNA.charAt(i+1))

+ String.valueOf(myDNA.charAt(i+2));

if(codon.equals("ATG")) {

for(int j = i; j < size - 2; j += 3) {

codon = String.valueOf(myDNA.charAt(j))

+ String.valueOf(myDNA.charAt(j+1))

+ String.valueOf(myDNA.charAt(j+2));

if (!(codon.equals("TAA") || codon.equals("TGA")

|| codon.equals("TAG"))) {

codonStr += codon;

}

if (codon.equals("TAA") || codon.equals("TGA")

|| codon.equals("TAG")) {

codonStr += codon;

return codonStr;

}

}

}

}

return codonStr;

}

The translateCodons() function takes in a codon string and returns an amino acid string. The function translate the codon using a series of if-else statements. Each codon is 3 characters. To translate the codon it takes the first character and compares it to either “A”, “C”, “T” or “G” then checks the second letter and compares, then check the last character and compares. Depending on the characters that had matched it produces a new character. It does this comparison or every 3 character of the whole codon string and produces a new translated string that will be the amino acid string.

static public String translateCodons(String myCodonStr) {

String aminoAcidSqn = "";

int size = myCodonStr.length();

int arySize = myCodonStr.length() / 3;

String[] codonAry = new String[arySize];

int k = 0;

for(int i = 0; i < arySize; i++) {

codonAry[i] = String.valueOf(myCodonStr.charAt(k))

+ String.valueOf(myCodonStr.charAt(k+1))

+ String.valueOf(myCodonStr.charAt(k+2));

k+=3;

}

int j = 0;

for(int i = 0; i < arySize; i++) {

if(codonAry[i].charAt(j) == 'A') {

if(codonAry[i].charAt(j+1) == 'A') {

if(codonAry[i].charAt(j+2) == 'A') {

aminoAcidSqn += "K";

} else if(codonAry[i].charAt(j+2) == 'C') {

aminoAcidSqn += "N";

} else if(codonAry[i].charAt(j+2) == 'G') {

aminoAcidSqn += "K";

} else if(codonAry[i].charAt(j+2) == 'T') {

aminoAcidSqn += "N";

}

} else if(codonAry[i].charAt(j+1) == 'C') {

if(codonAry[i].charAt(j+2) == 'A') {

aminoAcidSqn += "T";

} else if(codonAry[i].charAt(j+2) == 'C') {

aminoAcidSqn += "T";

} else if(codonAry[i].charAt(j+2) == 'G') {

aminoAcidSqn += "T";

} else if(codonAry[i].charAt(j+2) == 'T') {

aminoAcidSqn += "T";

}

} else if(codonAry[i].charAt(j+1) == 'G') {

if(codonAry[i].charAt(j+2) == 'A') {

aminoAcidSqn += "R";

} else if(codonAry[i].charAt(j+2) == 'C') {

aminoAcidSqn += "S";

} else if(codonAry[i].charAt(j+2) == 'G') {

aminoAcidSqn += "R";

} else if(codonAry[i].charAt(j+2) == 'T') {

aminoAcidSqn += "S";

}

} else if(codonAry[i].charAt(j+1) == 'T') {

if(codonAry[i].charAt(j+2) == 'A') {

aminoAcidSqn += "I";

} else if(codonAry[i].charAt(j+2) == 'C') {

aminoAcidSqn += "I";

} else if(codonAry[i].charAt(j+2) == 'G') {

aminoAcidSqn += "M";

} else if(codonAry[i].charAt(j+2) == 'T') {

aminoAcidSqn += "I";

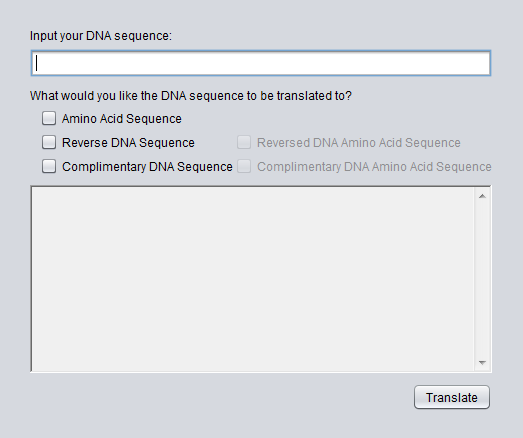
}

}

. . .

**User Interface Description:**

1. Enter DNA string in the “Input your DNA sequence:” box.
2. Check each box depending on what you want your DNA string to be translated into.
   1. You can check as many boxes as you like.
   2. In able to check the “Reversed DNA Amino Acid Sequence” box you need to first need to check the “Reverse DNA Sequence” box.
   3. In able to check the “Complimentary DNA Amino Acid Sequence” box you need to first need to check the “Complimentary DNA Sequence” box.
3. Click the “Translate” button.



**Sample Output:**

