// C++ Project Code - DNA transcription and translation ~ Jenny Lam

#include "stdafx.h"

#include <iomanip>

#include <iostream>

#include <cstdlib>

#include <string>

#include <conio.h>

#include <cmath>

#include <windows.h>

using namespace std;

void title();

void instructions();

string transcripDNA();

void codonTable(char amino\_acid[][4][4]);

string translateRNA(string);

void DNAstability(string);

void title()

{

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), 11); // light cyan text color

cout << "\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;

cout << "\t| | | | | | | | | | | | |" << endl;

cout << "\tA G C A T T G C A G C T A\n\n\n" << endl;

cout << "~~\t\t" << "D E C O D I N G D N A" << setw(20) << "~~" << endl;

cout << "\n\n\n\tT A G T A A C G T C G A T" << endl;

cout << "\t| | | | | | | | | | | | |" << endl;

cout << "\t\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n\n\n" << endl;

}

void DNAstability(string dna)

{

double count(0);

double CGpercentage;

std::cout << std::fixed;

std::cout << std::setprecision(4);

for (int i(0); i < dna.size(); i++) // loop to go through inputted dna strand

{

if (dna[i] == 'C' || dna[i] == 'G')

{

count += 1.0;

}

}

CGpercentage = (count / dna.length()) \* 100;

cout << "\nThe CG% is : " << CGpercentage << "%" << endl;

cout << "DNA Stability: ";

if (CGpercentage >= 75.0)

cout << " HIGHLY STABLE\n\n";

else if (CGpercentage >= 50.0 && CGpercentage < 75.0)

cout << " STABLE\n\n";

else if (CGpercentage >= 25.0 && CGpercentage < 50.0)

cout << " LESS STABLE\n\n";

else if (CGpercentage < 25.0)

cout << " UNSTABLE\n\n";

}

string transcripDNA(string dna) // transcribes DNA strand into corresponding nucleotide bases

{

string rna = "";

for (int i(0); i < dna.length(); i++) // dna transcription

{

if (dna[i] == 'A') { rna += 'U'; }

else if (dna[i] == 'C') { rna += 'G'; }

else if (dna[i] == 'G') { rna += 'C'; }

else { rna += 'A'; } // if dna[i] == 'U', add 'A' base to string rna

}

return rna; //newly transcribed dna

}

void codonTable(char amino\_acid[][4][4])

{

// declare 3d array for each base in a codon, each of element size 4 for each of the four bases ACGU.

//A=0, C=1, G=2, U=3

// phe - phenylalanine

amino\_acid[3][3][3] = 'F';

amino\_acid[3][3][1] = 'F';

// leu - leucine

amino\_acid[3][3][0] = 'L';

amino\_acid[3][3][2] = 'L';

amino\_acid[1][3][3] = 'L';

amino\_acid[1][3][0] = 'L';

amino\_acid[1][3][2] = 'L';

amino\_acid[1][3][1] = 'L';

// ile - isoleucine

amino\_acid[0][3][3] = 'I';

amino\_acid[0][3][1] = 'I';

amino\_acid[0][3][0] = 'I';

//met (Start codon) - methionine

amino\_acid[0][3][2] = 'M';

//val - valine

amino\_acid[2][3][3] = 'V';

amino\_acid[2][3][1] = 'V';

amino\_acid[2][3][0] = 'V';

amino\_acid[2][3][2] = 'V';

//ser - serine

amino\_acid[3][1][3] = 'S';

amino\_acid[3][1][1] = 'S';

amino\_acid[3][1][0] = 'S';

amino\_acid[3][1][2] = 'S';

//pro - proline

amino\_acid[1][1][3] = 'P';

amino\_acid[1][1][1] = 'P';

amino\_acid[1][1][0] = 'P';

amino\_acid[1][1][2] = 'P';

//thr - threonine

amino\_acid[0][1][3] = 'T';

amino\_acid[0][1][1] = 'T';

amino\_acid[0][1][0] = 'T';

amino\_acid[0][1][2] = 'T';

//ala - alanine

amino\_acid[2][1][3] = 'A';

amino\_acid[2][1][1] = 'A';

amino\_acid[2][1][0] = 'A';

amino\_acid[2][1][2] = 'A';

//tyr - tryptophan

amino\_acid[3][0][3] = 'Y';

amino\_acid[3][0][1] = 'Y';

//Stop Codon

amino\_acid[3][0][0] = '-';

amino\_acid[3][0][2] = '-';

amino\_acid[3][2][0] = '-';

//his - histidine

amino\_acid[1][0][3] = 'H';

amino\_acid[1][0][1] = 'H';

//gln - glutamine

amino\_acid[1][0][0] = 'Q';

amino\_acid[1][0][2] = 'Q';

//asn - asparagine

amino\_acid[0][0][3] = 'D';

amino\_acid[0][0][1] = 'D';

//lys - lysine

amino\_acid[0][0][0] = 'K';

amino\_acid[0][0][2] = 'K';

//asp - asparagine

amino\_acid[2][0][3] = 'D';

amino\_acid[2][0][1] = 'D';

//glu - glutamate

amino\_acid[2][0][0] = 'E';

amino\_acid[2][0][2] = 'E';

//cys - cysteine

amino\_acid[3][2][3] = 'C';

amino\_acid[3][2][1] = 'C';

//trp - tryptophan

amino\_acid[3][2][2] = 'W';

//arg - arginine

amino\_acid[1][2][3] = 'R';

amino\_acid[1][2][1] = 'R';

amino\_acid[1][2][0] = 'R';

amino\_acid[1][2][2] = 'R';

amino\_acid[0][2][0] = 'R';

amino\_acid[0][2][2] = 'R';

//ser - serine

amino\_acid[0][2][3] = 'S';

amino\_acid[0][2][1] = 'S';

//gly - glycine

amino\_acid[2][2][3] = 'G';

amino\_acid[2][2][1] = 'G';

amino\_acid[2][2][0] = 'G';

amino\_acid[2][2][2] = 'G';

}

string translateRNA(string rna)

{

bool readSequence = false;

string protein = ""; // declare blank string

char amino\_acid[4][4][4];

codonTable(amino\_acid); // references codon table function for corresponding amino acid

int base1, base2, base3;

for (int i(0); i < rna.length() - 2; i += 3) // loop starts at zero -- since we want to check three bases at a time (codon), increment by 3.

{

if (rna[i] == 'A')

{

base1 = 0;

}

else if (rna[i] == 'C')

{

base1 = 1;

}

else if (rna[i] == 'G')

{

base1 = 2;

}

else

{

base1 = 3;

}

if (rna[i + 1] == 'A')

{

base2 = 0;

}

else if (rna[i + 1] == 'C')

{

base2 = 1;

}

else if (rna[i + 1] == 'G')

{

base2 = 2;

}

else

{

base2 = 3;

}

if (rna[i + 2] == 'A')

{

base3 = 0;

}

else if (rna[i + 2] == 'C')

{

base3 = 1;

}

else if (rna[i + 2] == 'G')

{

base3 = 2;

}

else

{

base3 = 3;

}

if (amino\_acid[base1][base2][base3] == amino\_acid[0][3][2]) { readSequence = true; } // starts reading when codon reads AUG

else if (amino\_acid[base1][base2][base3] == amino\_acid[3][0][0] || // stops reading when codon reads UAA, UAG, OR UGA

amino\_acid[base1][base2][base3] == amino\_acid[3][0][2] ||

amino\_acid[base1][base2][base3] == amino\_acid[3][2][0])

{

readSequence = false;

}

if (readSequence) // if read sequence is true, then add this to protein variable

{

protein = protein + amino\_acid[base1][base2][base3] + " ";

}

else { continue; }

}

return protein;

}

int main()

{

title();

char run;

string dna;

bool b;

cout << "Reading DNA can be compared to reading a sentence.\nWhen entering your DNA strand,";

cout << "please include a start codon (TAC) & a stop codon (ATT, ATC, ACT).\n " << endl;

cout << "Enter DNA strand consisting of nucleotide bases ATGC (Spaces must be omitted): \n";

do {

getline(cin, dna); // user enters data

for (int i(0); i < dna.length(); i++) // checking if DNA strand contains the bases AGTC

{

dna[i] = toupper(dna[i]); // changes lowercase letters to uppercase

if (dna[i] == 'A' || dna[i] == 'G' || dna[i] == 'C' || dna[i] == 'T') // checks each element for correct base

b = true; // if true, loop runs again until it finds an incorrect base

else { b = false; break; } // invalid strand -> loop breaks for loop

}

while (!b) // once if-statement breaks, runs through loop to get a valid dna strand from user

{

cout << "Invalid entry. Re-enter DNA strand in terms of bases ATGC: \n";

getline(cin, dna);

for (int i(0); i < dna.length(); i++)

{

dna[i] = toupper(dna[i]);

if (dna[i] == 'A' || dna[i] == 'G' || dna[i] == 'C' || dna[i] == 'T')

b = true;

else

{

b = false; break;

}

}

}

cout << endl << endl;

cout << "||\t\t C E N T R A L\tD O G M A \t\t||" << endl;

cout << "\nDNA Strand:\n" << dna << endl << endl;

DNAstability(dna);

string rna = transcripDNA(dna); // stores transcribed DNA into string variable RNA

cout << "mRNA Strand: \n" << rna << endl << endl;

cout << "Amino Acid Sequence: \n";

string protein = translateRNA(rna);

cout << protein << endl;

cout << "Do you want to transcribe and translate another DNA strand? Enter Y/y for yes, N/n for no: \n";

cin >> run;

cin.ignore(1000,'\n'); //

} while (run == 'Y' || run == 'y');

system("pause");

return 0;

}