CIS1300 Assignment 2: genee

Goals of this Assignment

- Learn to create simple and nested loops
- Learn to create and use user-defined functions
- Learn to define and use one-dimensional arrays and strings
- Learn to read given requirements and convert them to C code
- Use problem solving skills to solve a given problem

1.0 Background Information

Bioinformatics is a field that studies the analysis and manipulation of complex biological structures such as DNA and Computer Science plays a large role in bioinformatics. In this assignment, you will use your problem-solving skills to write functions on DNA manipulation. There are 2 tasks you need to perform:

Task1: You are required to write function definitions for all functions defined in table 1. Each entry in the table has a function name, its prototype and its description.

Task 2: You must write a main program to test all functions that you write. However, DO NOT submit your main. When grading, we will test your functions from our main program.

Note: One approach to separate the above 2 tasks is to create 2 C files for this assignment, one named main.c that contains your main function (DO NOT submit this file), and the other lastnameFirstnameA2.c that has all function definitions (you MUST submit this file). The file named main.c must include all function prototypes, and the following header files:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

It (main.c) must also include the following constant definitions:

```
#define SIZE 100
#define NUMPROTEINS 64
```

If your code requires a header file that is not listed above, then you must include it in lastnameFirstnameA2.c (e.g., if you use a bool datatype, then #include <stdbool.h>). Note that lastnameFirstnameA2.c **must** contain all your function definitions.

To compile them together, use the following gcc command:

```
gcc -std=c99 -Wall lastnameFirstnameA2.c main.c
```

More on this can be found in the section on “Submission Instructions”.

Requirements Description: A complete description of DNA and its structure are beyond the scope of this assignment, here is a general idea. There are many sites where you can learn more on this, one of them is: https://opentextbc.ca/biology/chapter/9-1-the-structure-of-dna/

- **RQ1:** The building blocks of DNA are nucleotides, which are made up of three parts: a deoxyribose, a phosphate group, and a nitrogenous base. There are four types of nitrogenous bases in DNA: Adenine (A), guanine (G), cytosine (C) and thymine (T). DNA consists of a linear series of these four bases (synonymously called as neuclotide in this assignment). The order of these bases directs the cell as to which proteins to express and what form those proteins will take.
- **RQ2:** A DNA sequence consists of two strands wound around each other, with each strand held together by bonds between the bases. Adenine (A) and thymine (T) are complementary base pairs, and cytosine (C) pairs with guanine (G). An example of a base-paired DNA sequence is:

  GGATC
  CCTAG

  Each row is called a strand of DNA; the two rows together form a sequence. Note that 'CCTAG' is a complementary strand of 'GGATC'. Also note that given one strand, the other can be constructed easily.

- **RQ3:** We all know what a palindrome is – something that reads the same backwards as it reads forward. For example, 'MADAM', is a palindrome, since it reads the same forward and backward. Similarly, 'CIVIC', 2002, 'GGCCGG' are also palindromes.

- **RQ4:** The rule to check if a DNA sequence is palindromic is different from the definition of a palindrome given above – we will call it as dna_palindrome in this assignment. For a sequence to be considered as a palindrome, its complementary strand must read the same in the opposite direction [https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3602881]. In DNA terminology, palindromes are defined as a sequence of nucleotides that are followed by its complement sequence appearing in reverse order.

  For example, strand GGATCC can make a dna_palindrome. Let's understand how.

  DNA strand: GGATCC
  Complementary Strand: CCTAGG

  Does the complementary strand read the same backwards (i.e., from right to left)? Yes, it does, therefore GGATCC make a dna_palindrome

  Similarly, does GGCCGG make a dna_palindrome? The answer is No

DNA strand: GGCCGG
Complementary Strand: CCGGCC
Does the complementary strand read the same backwards (i.e., from right to left)? No, it Does, therefore GGCCGG does not make a dna_palindrome

- **RQ5:** Often times, mRNA are used along with DNA – while DNA stands for Deoxyribonucleic acid, RNA stands for ribonucleic acid; mRNA is a messenger RNA. Details on this are not in the scope of this assignment. In a simplified form, mRNA of a DNA strand is the complement strand of the DNA with each occurrence of the nucleotide T replaced with the nucleotide U.

- **RQ6:** mRNA can be translated to a sequence of Amino Acids by "breaking" the mRNA sequence into groups of three nucleotides (e.g., "UUG"). Each nucleotide triple translates to one of 20 amino acids. Most biologists remember the amino acids by their single-letter codes (e.g., "UUG" has a single-letter code 'L'). For your convenience, you are given a function definition for a function named getCode that returns the single-letter code for a 3-letter nucleotide triple. **Attached is a file called getCode.c that has the function definition of getCode. You may use this function to find the sequence of amino acids in a given dna strand.** For example, "CGTAGGCAT" translates to A S V (because its equivalent mRNA is "GCAUCCGUA" and the letter code for "GCA" is 'A', for "UCC" is 'S', for "GUA" is 'V'). What if the length of the strand is not a multiple of 3? In such cases, you may ignore the remaining bases in the given strand). For example, "CACGT" translates to V (because this DNA's mRNA is "GUGCA" and the letter code for "GUG" is 'V' – the remaining "CA" are ignored, since there are not enough bases remaining to make a triplet).

1.1 Defined constants:

```
#define SIZE 100

#define NUMPROTEINS 64
```

- SIZE is the max size of a sequence in this assignment.
- NUMPROTEINS is the maximum number of protein names used in this assignment.

  NUMPROTEINS is used in the provided c code that has the function definition of getCode.

2.0 Function prototypes and description:

1	**Prototype:** bool isBasePair (char neu1, char neu2); **Description:** Returns true if nucleotides represented by chars neu1 and neu2 form a base pair, and false otherwise. **Example:** isBasePair('A', 'T')) returns true whereas isBasePair('A', 'C')) returns false Read **RQ1** for more information on this
2	**Prototype:** bool isItaDnaSequence (char strand1 [SIZE], char strand2 [SIZE]); **Description:** Returns true if strands represented by strand1 and strand2 form a DNA, and false otherwise. Pre-condition: Assume that length of strand1 = length of strand2 **Example:** isItaDnaSequence ("CT", "TA") returns false; whereas isItaDnaSequence ("CT", "GA") returns true. Read **RQ1** for more information on this
3	**Prototype:** void reverse (char aStrand [SIZE]); **Description:** This function takes a strand and reverses it. **Example:** Assume that a string named aStrand initialized as “CATGG” is given as input to function reverse. The function replaces aStrand with the reverse of the given input, i.e., with “GGTAC”.
4	**Prototype:** void complementIt (char aStrand [SIZE]); **Description:** This function takes a strand and complements it. **Example:** Assume that a string named aStrand initialized as “CATG” is given as input to function complementIt. The function replaces aStrand with its complement, i.e., with “GTAC”. Read **RQ2** for more information on this
5	**Prototype:** bool isItPalindrome (char aStrand [SIZE]); **Description:** Returns true if the given string is a palindrome; false otherwise **Example:** isItPalindrome (“MADAM”) returns true; whereas isItPalindrome (“CATG”) returns false Read **RQ3** for more information on this
6	**Prototype:** bool isStrandDnaPalindrome (char aStrand [SIZE]); **Description:** Returns true if the given string is a dna_palindrome; false otherwise **Example:** isStrandDnaPalindrome (“AT”) returns true; whereas isStrandDnaPalindrome (“AG”) returns false Read **RQ4** for more information on this
7	**Prototype:** int howMany (char aStrand [SIZE], char neu); **Description:** returns the total number of neuclotides (nue) in the given strand (aStrand). **Example:** returns 4 if called as howMany (“GGCCGG", ‘G’);

8	**Prototype:** void dnaToMrna (char aSeq [SIZE], char mRNA [SIZE]); **Description:** This function takes one DNA sequence as a parameter and returns the corresponding mRNA sequence. **Example:** Assume that a string named aSeq initialized as “CATG” is given as input to function dnaToMrna. The function then stores “GUAC” in mRNA. Read **RQ5** for more information on this.
9	**Prototype:** void translateDnaToMrnaProteins (char aSeq [SIZE]); **Description:** This function takes a DNA sequence, and prints the list of amino-acids that its mRNA equivalent translates to. Note that you are given a function definition called getCode that you may use in your code, especially in this function. **Example:** If the given DNA sequence is “CGTAGGCAT”, then the amino acids that it translates to are ‘A”, ‘S’ and ‘V’; therefore your function prints DNA: CGTAGGCAT mRNA: GCAUCCGUA, which translates to: GCA : A UCC : S GUA : V Similarly, If the given DNA sequence is “CACGC”, then your function prints DNA: CACGC mRNA: GUGCG, which translates to: GUG : V And if the given DNA sequence is “CAXGC”, then your function prints DNA: CAXGC mRNA: GUXCG, which translates to: GUX : Z The input sequence has an incorrect base Read **RQ6** for more information on this

3.0 Submission Instructions:

- Submit a single C file containing your function definitions only (DO NOT submit main). To submit, upload your C file to the submission box for A2 on Courselink. Name your file as lastnameFirstnameA2.c (For example, if Ritu is the first name and Chaturvedi is the last name, the file would be called chaturvediRituA2.c). Incorrect file name will result in penalty.
- Incorrect format of submitted files will result in automatic zero. (Must be a valid .c file)
- The program you submit must compile with no warnings and run successfully for full marks. You get a zero if your program doesn't compile. There is also a penalty for warnings (5% for each unique warning).
- Penalties will occur for missing style, comments, header comments etc.
- DO NOT use global variables. Use of any global variables will result in automatic zero.
- DO NOT use goto statements. Use of any goto statements will result in automatic zero.
- Use the template given below for header comment.
  /!\ Note: The file name, student name and email ID must be changed per student.

```
    /************************chaturvediRituA2.c**************

        Student Name: Ritu Chaturvedi Email Id: ritu

       Due Date: November 3rd                Course Name: CIS 1300

       I have exclusive control over this submission via my password.
       By including this statement in this header comment, I certify that:

       1) I have read and understood the University policy on academic integrity. 2) I
       have completed the Computing with Integrity Tutorial on Moodle; and 3) I have
       achieved at least 80% in the Computing with Integrity Self Test.

       I assert that this work is my own. I have appropriately acknowledged any and
       all material that I have used, whether directly quoted or paraphrased.
       Furthermore, I certify that this assignment was prepared by me specifically for
       this course.

*****************************************************************************/
```

The program file must contain instructions for the TA on how to compile and run your program in a header comment.

/!\ Note: The file name must be changed per student.

```
/*****************************************************************
Compiling the program
The program should be compiled using the following flags: -std=c99 -Wall

compiling:
gcc -std=c99 -Wall chaturvediRituA2.c main.c

Running: ./a.out
OR
```

```
gcc -std=c99 -Wall chaturvediRituA2.c main.c -o assn2

Running the Program: ./assn2
*********************************************************/
```