CSCI 1300: Starting Computing
Instructor: Fleming
Homework #5

Deadlines:

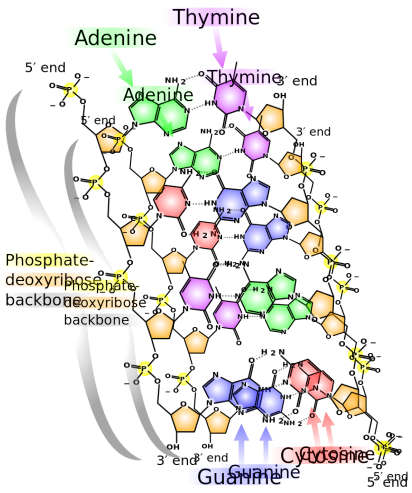
- You need to complete your submission by Saturday, October 13th at 6pm
- +5% by Thursday, October 11th at 11:55pm
- +2% by Friday, October 12th at 11:55pm

Objectives:

- Use string operations
- Use loops to go through strings
- Write test cases to test corner cases

Measuring DNA Similarity

DNA is the hereditary material in humans and other species. Almost every cell in a person's body has the same DNA. All information in DNA is stored as code in four chemical bases: adenine (**A**), guanine (**G**), cytosine (**C**) and thymine (**T**). The differences in the order of these bases is a means of specifying different information.



We refer to an organism's complete set of chemical bases as their *genome*. Every genome looks something like:

GATCAATGAGGTGGACACCAGAGGCGGGGACTTGTAAATAACACTGGGCTGTAGGAGTGATGGGGTTCACCTCTAATTCTAAG
ATGGCTAGATAATGCATCTTTCAGGGTTGTGCTTCTATCTAGAAGGTAGAGCTGTGGTCGTTCAATAAAAGTCCTCAAGAGGTTG
GTTAATACGCATGTTTAATAGTACAGTATGGTGACTATAGTCAACAATAATTTATTGTACATTTTTAAATAGCTAGAAGAAAAGCAT
TGGGAAGTTTCCAACATGAAGAAAAGATAAATGGTCAAGGGAATGGATATCCTAATTACCCTGATTTGATCATTATGCATTATATA
CATGAATCAAAATATCACACATACCTTCAAACTATGTACAAATATTATATACCAATAAAAAATCATCATCATCATCTCCATCATCAC
CACCCTCCTCCTCATCACCACCAGCATCACCACCATCATCACCACCACCATCATCACCACCACCACTGCCATCATCATCACCAC
CACTGTGCCATCATCATCACCACCACTGTCATTATCACCACCACCATCATCACCAACACCACTGCCATCGTCATCACCACCACTG
TCATTATCACCACCACCATCACCAACATCACCACCACCATTATCACCACCATCAACACCACCACCCCCATCATCATCATCACTAC
TACCATCATTACCAGCACCACCACCACTATCACCACCACCACCACAATCACCATCACCACTATCATCAACATCATCACTACCACC
ATCACCAACACCA

Human Genome: First 1000 lines of Chromosome 1 (http://www.sacred-texts.com/dna/hgp011k.htm)

The human genome has about 3 billion DNA base pairs. That means the entire human genome can be represented by a (very long) string of ~3 billion characters, where every character in the string is A, C, G, or T. You can find the first 1000 lines of Chromosome 1 of the Human Genome at this webpage.

One of the challenges in computational biology is determining where a DNA binding protein will bind in a genome. Each DNA binding protein has a preference for a specific sequence of nucleotides(basic structural unit for DNA). This preference sequence is known as a motif. The locations of a motif within a genome are important to understanding the behavior of a cell's response to a changing environment.

To find each possible location along the DNA, we must consider how well the motif matches the DNA sequence at each possible position. The protein does not require an exact match to bind and can bind when the sequence is similar to the motif.

Another common DNA analysis function is the comparison of newly discovered DNA genomic sequences to the large databases of known DNA sequences and using the similarity of the sequences to help identify the origin of the new sequences.

Hamming distance

Hamming distance is one of the most common ways to measure the similarity between two strings of the same length. Hamming distance is a position-by-position comparison that counts the number of positions in which the corresponding characters in the string are different. Two strings with a small Hamming distance are more similar than two strings with a larger Hamming distance.

Example: first string = "ACCT" second string = "ACCG"

```
A C C T
|   |   |  *
A C C G
```

In this example, each string has 4 characters

genomeLength = 4

there is 1 mismatch pair of characters, so:

hammingDistance = 1

Similarity score :
Similarity score defines how similar first string is compared to the second string. :

The *similarity score* for two sequences is then calculated as follows:

similarityScore = (genomeLength - hammingDistance) / genomeLength

similarityScore = (4-1)/4 = 3/4 = 0.75

Two sequences with a high similarity score are more similar than two sequences with a lower similarity score. The two strings **must always be the same length** when calculating a Hamming distance. If the length of the genomes is different, then the similarity score cannot be calculated. Similarity score will be in the range [0.0,1.0].

QUESTION:
Suppose that the emergency room of some hospital sees a sudden and drastic increase in patients presenting with a particular set of symptoms. Doctors determine the cause to be bacterial, but without knowing the specific species involved they are unable to treat patients effectively. One way of identifying the cause is to obtain a DNA sample and compare it against known bacterial genomes. With a set of similarity scores, doctors can make more informed decisions regarding treatment, prevention, and tracking of the disease. For your homework, you will build a program to find the best matching genome sequence for a given bacterial sequence.

Specifications

- Your solution must have an algorithm in pseudocode explaining how you are approaching the problem, step by step.
- Your program needs to have `simScore()` and `analyzer()` functions
- You need to create at least 1 functions of your choice.
- In the CPP file, you must have at least 6 test cases for each function, testing general cases and corner/edge cases. More details at the end of this write-up, coffeeCost.cpp example.

Required Function 1: Find Similarity Score

Write a function that returns the similarity score for two sequences. The similarity score for two sequences is calculated as follows:

similarity_score = (string length - hamming_distance) / string length

where hamming distance is a position-by-position comparison that counts the number of positions in which the corresponding characters in the string are different. Two strings with a small Hamming distance are more similar than two strings with a larger Hamming distance.

Example: first string = “ACCT” second string = “ACCG”

A C C T

| | | *

A C C G

In this example, there are 3 matching characters and 1 mismatch, so: hamming_distance = 1 . And the similarity score for the two strings on this example would be, similarity_score = (4-1)/4 = 3/4 = 0.75

- Your function MUST be named **simScore**
- Your function should take two parameters in this order:
  - a string parameter for the first sequence
  - a string parameter for the second sequence.
- Your function should return the similarity score as a double.
- Your function should not print anything.

Edge Cases:

1. If the lengths of sequences are different → return 0
2. If either sequence is an empty string → return 0

Examples:
Arguments and their expected return values for the simScore

sequence1	**sequence2**	**Similarity score**
ATGC	ATGA	0.75
CCDCCD	CCDCCD	1

ATG	GAT	0
ACCDT	ACCT	0
CGT	DGGTA	0

Required function 2: Finding the best DNA match

Write a function that will help the doctors find the best matching amongst the three known genomes with a sequence from the unknown bacteria.

- The function must be named **analyzer**
- The function does not return any value.
- The function will take in 4 string input parameters in the following order:
    - The first string parameter corresponds to genome 1
    - The second string parameter corresponds to genome 2
    - The third string parameter corresponds to genome 3
    - The fourth string parameter corresponds to the sequence of the unknown bacteria

- The function should display the best matching genome amongst the three genomes:
    - Eg : If genome 1 is the best match, the output would be
        - **Genome 1 is the best match.**

- If 2 or more genomes are the best matches, then display all the best matches in separate lines:
    - E.g. If genome 2 and 3 are the best matches then display the following:
        - **Genome 2 is the best match.**
          Genome 3 is the best match.

- If either of the genomes or sequence is empty, display the following output and exit the function:
    - **Genome and sequence cannot be empty.**

- **Note: The length of the three genomes should be the same.** If the three genome length does not match, display the following output and exit the function:
    - **Genome length does not match.**

- If the sequence length is greater than any of the genome, display the following output and exit the function:
    - **Sequence length must be smaller than genome length.**

Example1: One of the genomes has the best match

genome 1	AATGTTTCAC
genome 2	GACCGACTAA
genome 3	AAGGTGCTCC
sequence	TACTA
Genome 2 is the best match.	

Explanation:
Since the length of the three genomes (genome1, genome3, and genom3) are the same, we calculate the best similarity score in each sequence. The best-matched genome is calculated based on the highest similarity scores given in each sequence.

		Highest similarity score
genome 1	AATGT**TTCAC**	0.4
genome 2	GACC**GACTA**A	0.8
genome 3	AAGG**TGCTC**C	0.6

The best similarity score for genome1 and sequence is 0.4, for genome2 and sequence is 0.8, and for genome3 and sequence is 0.6. Since genome 2 has the highest similarity scores among the three genomes, genome 2 is best matched.

Example2: Two genomes match with a sequence

genome 1	AACT
genome 2	AACT
genome 3	AATG
sequence	AACT
Genome 1 is the best match. Genome 2 is the best match.	

Explanation:
The best-matched genome is calculated based on the highest similarity scores given in each sequence.

		Highest Similarity Score
genome 1	**AACT**	1.0
genome 2	**AACT**	1.0
genome 3	AATG	0.5

The length of the three genomes is the same, so we calculate the best similarity score. The best similarity for genome1 and genome2 will be 1.0, while the genome3 is 0.5. Since genome 1 and 2 have higher similarity scores than the genom3, it will be printed in the format as specified.

Example3: three matched sequence

genome 1	**ACAT**C
genome 2	**ACTT**A
genome 3	T**ACAT**
sequence	AACT
Genome 1 is the best match. Genome 2 is the best match. Genome 3 is the best match.	

All of the three genomes have the same number of characters, so we check the best similarity score in each genome. Since all genome has the same similarity score of 0.5, so the output shows all three genomes as the best match.

Example4: Genome and sequence cannot be empty

genome 1	
genome 2	CATTA
genome 3	TAATC
sequence	ACTA
Genome and sequence cannot be empty.	

Explanation:
Since genome1 is an empty string, your program outputs “Genome and sequence cannot be empty.”

Example5: Three genome length does not match

genome 1	TAATC
genome 2	CATTAA
genome 3	TAATC
sequence	ACTA
Genome length does not match.	

Explanation:
Genome2 is longer than the others. The program just outputs an error message, “Genome length does not match.”

Example6: the sequence length is greater than any of the genome

genome 1	TAATC
genome 2	CATTA
genome 3	TAATC
sequence	ACTATTT
Sequence length must be smaller than genome length.	

Explanation:
Sequence is longer than the others. The program just outputs an error message, “Sequence length must be smaller than genome length.”

How to write test cases (You need to write test cases for each function you create)

```
// example from lecture, coffeeCost.cpp (posted on Moodle)

// invalid input, then return -1
double CoffeeCost(int cups, double price)
{
    double totalPrice = -1;
    if(cups >= 0 && price >= 0) // input is valid
    {
        int rewards = cups/13; // how many free cups
        if (rewards >= 1) // have free coffee
        {
            totalPrice = (cups - rewards) * price;
        }
        else
        {
            totalPrice = cups * price;
        }
    }
    return totalPrice;
}

int main()
{
    // tests for CoffeeCost(...)
    cout << CoffeeCost( 4, 5) << endl; // expected value: 20
    cout << CoffeeCost( 11, 5)<< endl; // expected value: 55
    cout << CoffeeCost( 12, 5)<< endl; // expected value: 60
    cout << CoffeeCost( 13, 5)<< endl; // expected value: 60
    cout << CoffeeCost( 14, 5)<< endl; // expected value: 65
    cout << CoffeeCost( 24, 5)<< endl; // expected value: 115
    cout << CoffeeCost( 25, 5)<< endl; // expected value: 120
    cout << CoffeeCost( 26, 5)<< endl; // expected value: 120
    cout << CoffeeCost( 27, 5)<< endl; // expected value: 125
    cout << CoffeeCost( -26, 5)<< endl; // expected value: -1
    cout << CoffeeCost( 27, -5)<< endl; // expected value: -1
    cout << CoffeeCost( -26, -5)<< endl; // expected value: -1
    return 0;
}
```

Grading Rubric:

Criteria	Points
Coderunner Algorithm / Pseudocode Comments and explanation Test cases (at least 6 test cases per function)	75 5 5 15
Total Early submission bonus	100 + 5% or 2%