Humans vs Neanderthals

A bioinformatic quest

Your friends from the anthropology lab asked you to help. They excavated a site full of remains with the age of approximately 40,000 years. The bones suggested that they belonged to modern humans (*Homo sapiens sapiens*) or Neanderthals (*Homo sapiens neanderthalensis*), but it is impossible to distinguish between the two based on the visual features.

Mitochondrial DNA (mtDNA) was extracted from the bones and sequenced, and 11 different samples were found. You are given mtDNA references for human and neanderthal specimens, and **your goal is to divide 11 samples into humans** whose mtDNA is more related to the human reference **or Neanderthals** which are closer to the Neanderthal reference.

DNA, short for deoxyribonucleic acid, is the molecule that contains the genetic code of organisms. It consists of two strands build from four nucleotides, adenine (A), thymine (T), cytosine (C), and guanine (G). DNA can be presented as a string of four letters of one strand, for example, "ACCTGTG", and the second DNA strand can be easily restored from this string.

Part A.

First, you need to prepare four functions which will help you to complete the quest.

- **gc_content (string dna)**

This function takes a string of DNA sequence and calculates the percentage of letters G and C in this string. Thus, the GC content of "ACTG" is 50%.

- **hamming_distance (string dna1, string dna2)**

The Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different. For example, the Hamming distance between "AAAA" and "AAAC" is 1, "AAAA" and "ACTG" is 3, "AAAA" and "AAAA" is 0.

- **reverse_complement (string dna)**

Reverse Complement converts a DNA sequence into its reverse-complement counterpart. You may want to work with the reverse-complement of a sequence, because it may hide the information that interests you on the reverse strand. The reverse strand contains complement nucleotides, where A is always paired with a T, T with an A, G with a C, and C with a G, and is correctly presented from right to left. For the string "AACC", the reverse-complement is "GGTT", because you start from the right and pair C with G and A with T. The reverse-complement string for "TAAACCTG" is "CAGGTTTA".

- **find_substring (string reference, string dna)**

DNA reference you are given maybe shorter than a DNA sample. Thus, you have to find a substring in the DNA sample which has the minimal Hamming distance with your reference. Do not forget about a reverse strand, maybe the best match is hidden there! Let's find the best match for the reference "AAAC" in the sequence "AAAGGTTTC". The Hamming distance between the first four letters "AAAG" and the reference is 1 (that is good), but there is a match with the minimal Hamming distance equal to 0 on the opposite strand, the reverse-complement substring of "GTTT". Thus, the distance between the reference and this sample is 0, and the best substring after the reverse-complement is "AAAC".

Define fuctions declared in "main.hpp" and successfully run
make demo && ./demo

Part B.

There are two built-in functions in the file "sequence.hpp" declared like:
map <string, string> getReference ( ); // returns a map containing two references
map <string, string> getSequences ( ); // returns a map containing 11 mtDNA samples

The references are "human" and "neanderthal" and their DNA sequences consist of 180 nucleotide letters. Each of 11 samples has a name as a key and a mtDNA sequence (>= 180 letters) as a value. There are examples of how to access these pairs in "Demo.cpp".

You need to find and compare minimal Hamming distances between a sample and each of two references, when identify the sample as "human" or "neanderthal" according to the smaller value that biologically means this sample more likely belonged to a human or a Neanderthal. In case when the

sample has an equal minimal Hamming distance with both references, you have to calculate a GC content of this sample and group the sample with the reference whose percentage of the GC content is closer.

Combine functions you created in Part A and use them in the final quest.

- **quest ( )**

This function does not take any arguments and must return a string of keys organized exactly in this order: keys of human mtDNA samples sorted alphabetically and separated by a single space, then a semicolon, a space, and keys of Neanderthal mtDNA samples sorted alphabetically and separated by a single space.

For example,
humans: AB10.1 AA3.2
neanderthals: AA5.6 SD700.4
the only one correct answer: "AA3.2 AB10.1; AA5.6 SD700.4"

Define the fuction "quest" declared in "main.hpp" and successfully run
make test && ./test

Good luck!