Version 1.02

Assignment 02
DNA Editing

1. Submission Guidelines

Deadline:	16:00 PM on Monday 23 August 2021
Submission procedure:	Submit only one file labelled a2.cpp on blackboard (via TurnItIn)
Compilation:	Your code must compile using... `g++ a2.cpp tictoc.cpp -std=c++11` with either MinGW on PC or Terminal on MAC as described in Lecture 1.

2. Overview

Write C++ code that loads DNA in the FASTA format (`.fna`) and allows the user to analyse and edit them.

Throughout this assignment, gracefully handle both good and bad user inputs. For example, we may present the user with a menu of options 1 through 4. If the user enters a value between 1 and 4, then process the request accordingly. However, if the user does not enter a value between 1 and 4, or enters a bizarre entry such as a 'q' or "alligator", then redisplay the menu and ask the user to select an option again.

3. Description

Code a menu for the user to navigate. At the highest level, have four options:

```

Welcome to the DNA Editing program

Select an option:
(1) Load DNA(s).
(2) Process a DNA.
(3) Analyse the DNA database
(4) Quit.
>
```

Option 1. Load DNA(s)

If option 1 is selected, ask the user for one or more file names.

```
Enter the DNA file names:
For multiple files, separate them by a comma. Only .fna are recognised.
>
```

Handle the user's input gracefully, removing any white space before the first character of a file name and removing any white space after the final character of a file name. Spaces between the first and last characters of the file name are allowed. Once the file names are gathered, load the DNA sequences from the correctly labelled file names (`.fna` files) into the DNA database (see below). Do not load any files that are not appropriately named.

Option 2. Process a DNA

Ask the user to select a DNA sequence to process. The DNA should be presented in the order that they were loaded.

```
Select one of the DNA to process:
(1) example_dna_1.fna
(2) example_dna_2.fna
(3)
```

...

```
(N) example_dna_N.fna
```

```
>
```

Present the user with a sub-menu of options for processing this DNA sequence. Repeat this menu until option 9 is selected.

```
Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>
```

For option 1, ask the user to enter a DNA sequence on the console. Find that DNA sequence in the DNA sequence currently being analysed. Print back a list of matches that include (1) the base pair position (first and last sequence index), (2) base pair length, (3) 10 base pairs preceding the sequence, (4) the matching sequence, and (5) 10 base pair located after the sequence.

For option 2, perform the same process as option 1, except, ask the user for a filename with a `.fna` extension.

For option 3, ask the user to enter (1) a DNA sequence on the console and (2) the base pair position to add the sequence. Update the DNA sequence currently being analysed so that the new sequence is inserted at the base pair position. For example, if the DNA sequence is GTCAGTCA, then adding TGA at base pair position 2 will result in GTTGACAGTCA. After adding the DNA sequence, print (1) the base pair positions (first and last sequence index), (2) base pair length, (3) the 10 base pairs preceding the sequence added, (4) the sequence added, and (5) 10 base pairs located after the sequence added.

For option 4, perform the same process as option 3, except, ask the user for a filename with a `.fna` extension.

For option 5, ask the user to enter (1) a base pair position and (2) a base pair length. Print (1) the 10 base pairs preceding the sequence to be deleted, (2) the sequence to be deleted, and (3) 10 base pair located after the sequence. Notify the user that the sequence has been deleted. Print the (1) 10 base pairs preceding the deleted region and (2) 10 base pairs located after the deleted sequence.

For option 6, ask the user to enter (1) a nucleotide sequence on the console, (2) a base pair position, and (3) a base pair length. Replace the DNA sequence defined by the base pair position and length with the user-inputted nucleotide sequence. Print (1) the base pair positions (first and last sequence index), (2) base pair length, (3) the 10 base pairs preceding the sequence added, (4) the sequence added, and (5) 10 base pairs located after the sequence added.

For option 7, perform the same process as option 6, except, ask the user for a filename with a `.fna` extension.

For option 8, ask the user for a file name and save the edited DNA sequence.

For option 9, exit this sub menu.

Option 3. Analyse the DNA database

Ask the user for a filename with a `.fna` extension. Using the DNA sequence read from that filename, search all DNA sequences in the database for a match. Print back a list of matches that include (1) the DNA file name it was found in, (2) the base pair position (first and last sequence index), (3) base pair length, (4) 10 base pairs preceding the sequence, (5) the matching sequence, and (6) 10 base pairs located after the sequence.

Option 4. Quit

When selected, quit the program.

4. Coding Rules

In this assignment, you must use object-oriented programming and linked lists as described below. In order for us to assess whether you understand these two concepts, we have also listed several restrictions, limiting where you may use `vector`, `string`, and other C++ standard libraries.

Object-oriented programming

At the top of your `main()` function, declare an object `dna_db` of type `DNADatabase`, a class type that you are tasked to define.

```
int main()
{
     DNADatabase dna_db;

     // your code

     return 0;
}
```

All loading, analysis, and processing of DNA sequences must be performed through member functions and operators defined in `DNADatabase`. When coded appropriately, `friend` functions and operators are allowed. Thus, in your `main()` function, there should be no loading, analysis, or processing of DNA sequences without the use of `dna_db`. In summary, use functions and operators associated with `dna_db` in the spirit of object-oriented programming.

Linked lists

The `DNADatabase` should store the DNA sequences ('G', 'C', 'T', 'A', etc.) in your own, custom-designed linked-list data structure. This means that when working with DNA sequences, you must NOT use `vector`, `string`, or any other data structures that are part of the C++ standard library (eg, no C++ standard template library). The one and only exception to using `string` with a DNA sequence is when you receive the input from the user (options 1, 3, and 6 of Option 2. Process a DNA). However, you should typecast the string into a `char*` or another data type before passing it into the `DNADatabase` object. This restriction from using `vector` and `string` is because we want you to code your own algorithms, such as searching, inserting, etc, rather than use built-in functions. While we are denying you from using any data structure that is part of the C++ standard library, you may use data structures that are part of the C++ language (e.g., static and dynamically allocated arrays).

When processing DNA sequences from a file, the DNA sequence that is being extracted from that file does not have to be stored as a linked list. However, you may not use `string`, `vector`, or any other data structure that is part of the C++ standard library.

When the data being modified is not a DNA sequence, C++ standard libraries including `vector` and `string` are allowed.

Efficiency

Since we will be marking for coding efficiency, we ask that you print out the time duration it takes for some operations to perform.

Please follow these instructions very carefully:

- Place `tictoc.h` in your folder containing `a2.cpp`
- Place `tictoc.cpp` in your folder containing `a2.cpp`
- Add `#include "tictoc.h"` at the top of your code in `a2.cpp`
- Use the following line to compile your code:

```
g++ a2.cpp tictoc.cpp -std=c++11
```

In your code, declare a variable named `time` of type `TicToc` at the top of your `main()` function. This class is defined in the `tictoc.h` and `tictoc.cpp` files. Do not modify these files as you will not be asked to send them to us.

```
int main()
{
  TicToc time;

  // rest of your code

  return 0;
}
```

You can start the timer by calling `time.tic()` and stop the timer by calling `time.toc()`. You can then print out the time elapsed by using the extraction operator << (eg, `cout << time << endl;`).

We want you to report the time elapsed for Option 1, loading the .fna files, Option 2's sub-options 1 through 8, and Option 3. Make sure to place the `tic()` function call after the last user input is captured for that specific option request. Otherwise, the time elapsed will contain the duration the user took to enter the input.

```
// receive the final user input needed to process the request

time.tic();

// perform the core procedures of this call, such as
// loading, adding, deleting, replacing, etc.

time.toc();
cout << endl;
cout << time << endl;
```

This will result in an output in seconds:

```
(time elapsed: XXXX.XXXX s)
```

5. The FASTA format

All DNA files will be in the FASTA format and the .fna file extension. For this assignment, you only need to consider nucleotides. Amino acids do not need to be accounted for.

FASTA format details:

https://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastDocs&DOC_TYPE=BlastHelp

Your program must account for many different types of DNA files. We will upload example .fna files on blackboard, including the human genome.

Additional DNA sequences may be found online:

https://www.ncbi.nlm.nih.gov/datasets/

Valid assumptions: Only one DNA sequence per FASTA format

6. Guidance and Marking

This is a very difficult assignment. My suggestion is to start writing the menu, which will be marked separately from the rest of the implementation and will allow you to be awarded partial credit. Please make sure your code compiles as it is very difficult to find partial credit when we cannot run the program. The following qualities will be marked:

(1) Ability to submit code via TurnItIn

(2) Understanding of C++ basics

(3) Implementation of a menu

(4) Implementation of object-oriented programming, linked lists, and the DNA portion of the assignment.

(5) Coding efficiency (this includes both run-time speed and unnecessary code)

(6) Commenting and coding style

Quality number 4 will have a large amount of marks.

6. Cheating

The code that you submit for assessment must be your own work.

When discussing programming with others, never send or receive code from another person if it is specifically designed for this assignment. Do not write the assignment together with another person. Instead, discuss syntax, mechanics, and how to approach the problem.

Example Output

```
Welcome to the DNA Editing program

Select an option:
(1) Load DNA(s).
(2) Process a DNA.
(3) Analyse the DNA database
(4) Quit.
>1

Load DNA strands

Enter the DNA file names:
For multiple files, separate them by a comma. Only .fna are recognised.
>chr22.fna, gnFOXF1.fna

Loading file #1 "chr22.fna"...
Loading file #2 "gnFOXF1.fna"...

(time elapsed: XXXX.XXXX s)

Select an option:
(1) Load DNA(s).
(2) Process a DNA.
(3) Analyse the DNA database
(4) Quit.
>2

Process a DNA

Select one of the DNA to process:
(1) chr22.fna
(2) gnFOXF1.fna
>2

Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>1

Find DNA sequence by input

Enter the DNA sequence to search (eg, GTCACT):
>TGGAGGGC

Match #0
base pair positions: [605:612]
base pair length:    8
prev 10 base pairs:  AGCCTGGCGC
region of interest:  TGGAGGGC
next 10 base pairs:  GGCCTGGGCA

Match #1
base pair positions: [1434:1441]
base pair length:    8
```

```
prev 10 base pairs:  TCAGGCTGCC
region of interest:  TGGAGGGC
next 10 base pairs:  TGCCTCTGCC

(time elapsed: XXXX.XXXX s)

Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>3

Add DNA sequence by input

Enter the DNA sequence to add:
>helloworld

Enter a base pair position:
>123

base pair positions: [123:132]
base pair length:    10
prev 10 base pairs:  CGGCCATGGA
region of interest:  helloworld
next 10 base pairs:  CCCCGCGTCG

(time elapsed: XXXX.XXXX s)

Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>5

Delete DNA sequence by input

Enter a base pair position:
>123

Enter a base pair length:
>5

DNA sequence deletion information:
base pair positions: [123:127]
base pair length:    5
prev 10 base pairs:  CGGCCATGGA
region of interest:  hello
next 10 base pairs:  worldCCCCG

The DNA sequence has been deleted.

DNA sequence deletion result:
```

```
base pair position:  123
prev 10 base pairs:  CGGCCATGGA
next 10 base pairs:  worldCCCCG

(time elapsed: XXXX.XXXX s)

Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>6

Replace DNA sequence by input

Enter the DNA sequence to add:
>ba

Enter a base pair position:
>123

Enter a base pair length:
>3

base pair positions: [123:124]
base pair length:    2
prev 10 base pairs:  CGGCCATGGA
region of interest:  ba
next 10 base pairs:  ldCCCCGCGT

(time elapsed: XXXX.XXXX s)

Select from one of the following options
(1) Find DNA sequence by input
(2) Find DNA sequence by file
(3) Add DNA sequence by input
(4) Add DNA sequence by file
(5) Delete DNA sequence by input
(6) Replace DNA sequence by input
(7) Replace DNA sequence by file
(8) Save edited DNA sequence
(9) Exit submenu
>9

Select an option:
(1) Load DNA(s).
(2) Process a DNA.
(3) Analyse the DNA database
(4) Quit.
>4

Exiting program.
```