BME506
Fall 2023 - Lab 1
Language Basics – DNA Manipulation

Duration: one week.

Note:

1. Every lab assignment must be done individually.
2. When you name a folder or a file, you **should** avoid spaces in those names. For example, if you need to name a folder as **GreenApple**, you should name it as **GreenApple** instead of **Green Apple.** Naming a folder or a file without spaces in the names is to avoid compilation error that may occur while using MinGW build tools (For details, you may see here: http://www.mingw.org/wiki/Getting_Started).
3. **ALL code should be compiled and run on the laboratory computers before submission.**

Goals:

In the following lab, you will continue to work with refining your command of core C/C++ programming constructs, paying particular attention to the appropriate use of functions, types and control flows.

Please indent your code properly, include appropriate comments, and use proper names for functions and variables.

Background:

Deoxyribonucleic Acid (DNA) is a molecule that encodes the genetic blueprint from which cells generate products essential in the development of living organisms. DNA is composed of two molecular strands that coil to form a double helix, and are often represented as a string of characters (each representing a fundamental component in the strand known as a nucleotide). There are 4 types of nucleotide: Adenine (A), Guanine (G), Thymine (T) or Cytosine (C). Thus DNA sequences can be represented as a string of these characters: e.g. "AGTCATTAGCC …"

In this lab, we will construct a series of functions that:

1. Allow a user to enter a DNA string
2. Validate whether the DNA string contains only valid nucleotides
3. Find the compliment of a valid DNA string

- The prototypes of the three functions mentioned above should be in a header file named **Lab1.h.**
- The implementation of the functions should be in a file named **Lab1.cpp.**
- The function **main** should also be in **Lab1.cpp.**
- The code of **Lab1.h** and the implementation of **main** (that should go in **Lab1.cpp**) has already been provided in the **Appendix** at the end of this file. You **must** use them in your implementation.

General steps to build the C++ project in Netbeans for Lab1:

1. Create a Netbeans project named **Lab1Proj** while taking care of the NOTE below.
   NOTE: While creating the project, please **make sure**
   - you set the field "**Project Location:**" appropriately. **Be sure to remember this project location** path. (To keep track of your **Project Location** you may do as follows: Once you have created the project, open a File Explorer, and go into your **Project Location** folder).
   - once you have chosen your **Project Location**, enter the word **Lab1Proj** in the field "**Project Name:**". Notice that your project name gets appended to your **Project Location** path.
   - you **UN-CHECK** the checkbox on the left of "**Create Main File**" (We do so since we do not need any file named main.cpp file in this project).
   - you Click **[Finish]** to create the project.
   - a new folder named **Lab1Proj** has got created automatically under your **Project Location** folder. This **Lab1Proj** folder is your Netbeans ***project folder***.
2. Under netbeans project **Lab1Proj**, right-click on the "Header Files" > "New" > "C++ Header File …". Set the field **"File Name:"** as **`Lab1`**. Set the **"Extension:"** as "**`h`**". Click **[Finish]**.
3. The file **`Lab1.h`** is created. Copy the content for the header file from Appendix into this newly create file.
4. Under netbeans project **Lab1Proj**, right-click on the "Source Files" > "New" > "C++ Source File …". Set the field **"File Name:"** as **`Lab1`**. Set the **"Extension:"** as "**`cpp`**". Click **[Finish]**.
5. The file **`Lab1.cpp`** is created. Copy the content for the source file from Appendix into this newly create file.
6. In **Lab1.cpp**, add the relevant functions that are intended to solve problems of **Part I**, **Part II**, and **Part III** given below.
7. Note: At this point, make sure that your two newly created files **`Lab1.h`** and **`Lab1.cpp`** exists in the **Lab1Proj** folder.
8. Right-Click on project **Lab1Proj** >
   Select "Properties" >
   Click "Run" >
   On right, choose "Console Type" to be "Standard Output"
   Click [Apply]
   Click [OK]
9. Build and run **Lab1Proj** project.

Important Notes:

1. When your code is error-free, it should build successfully (as seen at the compilation process at Netbean's "Output" window). **However,** at times, in the Netbean's *source editor* window, you may notice red line below a built-in method, such as `size()` method of built-in type string. You may notice this even if your program has built successfully—For example, in the invocation `str1.size()`, you may see a red line below the word `size`. This is misleading. Such a red line should disappear if you perform the following steps:

   Right-Click on project **Lab1Proj** >
   Select "Code Assistance" >
   Select "Clean C/C++ cache and restart IDE"

2. **If you ever want to remove a project from Netbeans,** right-click on project > click **"Close". AVOID** using **"Delete"** to remove a project from the Netbeans. If you ever remove a project using "Delete", you will have to create a new project all over again to replace the "deleted" project.

PART I: Entering a DNA string

1. Write a C++ function that allows the user to enter a string of characters (which may or may not represent a valid DNA string). Upon pressing <enter> the string should be read from standard input (`cin`) and returned by the function.

2. You should assume that `main()` calls the function, passing no args. The return type of the function should be `string`. The function prototype should be like:

```
string enterDNAString();
```

3. The program should echo the string entered, followed by the size of the string (number of characters) in brackets, for e.g. if the user enters "ABHDKDJ", the program will output:

```
ABHDKDJ (7 chars)
```

*** Note: the <iostream> library is all that is needed to use the "string" type. We can test string variable's (e.g. string aStr) length by the following:*

```
string aStr = "I am a string value";
cout << aStr.size();          // size() returns unsigned int
```

PART II: Validate the DNA string

1. Write a C++ function to check that the DNA string entered using Part I is valid – i.e. the string only contains valid nucleotides (A|G|T|C). These can be repeated any amount of time, but can only be these characters. There should also be no white spaces in the entered text.

2. The function should accept the original string variable (from Part I) as an argument (passed by value) and should return true/false as the result (true if valid). The function prototype should be:

```
bool checkValidity(string s);
```

3. The output to the screen should extend the output of Part I (as in the following examples):

```
ABHDKDJ (7 chars) : INVALID DNA
AGTC (4 chars) : VALID DNA
AgtCCcTa (8 chars) : VALID DNA
```

PART III: Find the compliment of a valid DNA string

1. Write a C++ function to output the compliment of a valid DNA string. The compliment of each nucleotide is:

   A ⇔ T and G ⇔ C

   So in locations in the original sequence, A's will become T's, while T's will become A's. Similarly, G's will become C's, and C's will become G's

2. The function should accept a DNA string (from Part I) as an argument (passed by value). The return type of the function should be `void`. If the DNA string is invalid, then no compliment will be calculated. The function prototype should be:

```
void compliment(string s);
```

3. The output to the screen should extend the output of Part II (as in the following examples):

```
ABHDKDJ (7 chars) : INVALID DNA
AGTC (4 chars) : VALID DNA : compliment(AGTC) = TCAG
AgtCCcTa (8 chars) : VALID DNA : compliment(AgtCCcTa) = TcaGGgAt
```

Lab Submission

Deadline: See announcement in D2L for deadline.

ALL code should be compiled and run on the laboratory computers before submission.

Create a folder. Name it as YourLastname_YourFirstname_bme506_Labnumber.
Example: If student name is John Smith, the name of folder should be Smith_John_bme506_Lab1.

Copy your Netbeans ***project folder***, **Lab1Proj**, in the above folder. Make sure the folder **Lab1Proj** contains your two files **`Lab1.h`** and **`Lab1.cpp`**.

The above folder must also contain a duly filled and signed standard cover page. The cover page can be found on the departmental web site:
Standard Assignment/Lab Cover Page

Compress the above folder as a single zip file that is named according to the following rule:
YourLastname_YourFirstname_bme506_Labnumber.zip.
Example: Smith_John_bme506_Lab1.zip.

Upload the above zip file on D2L through the "Assessment" > "Assignments" link.

Note: If the code does not compile, the submission will receive a ZERO mark.

Resources

[1] Cplusplus.com Reference [Online]. Available: http://www.cplusplus.com/reference

Appendix

In your Lab1.h file, the code should be as follows:

```
#ifndef LAB1_H_
#define LAB1_H_

#include <iostream>
#include <string>
#include <climits>  //for INT_MAX

using namespace std;

string enterDNAString();
bool checkValidity(string s);
void compliment(string s);

#endif /* LAB1_H_ */
```

In your Lab1.cpp file, the #include statement and the function `main` should be as follows. PLEASE DO NOT CHANGE THE BODY OF `main`.

```
#include "Lab1.h"

//implement the function enterDNAString
//implement the function checkValidity
//implement the function compliment

int main() {

    while(true) {
        cout << "Please choose an option:" << endl;
        cout << "1. Enter DNA string, echo it" << endl;
        cout << "2. Enter DNA string, echo it, validate it" << endl;
        cout << "3. Enter DNA string, echo it, validate it, compliment it if valid"
             << endl;
        cout << "4. Exit" << endl;

        int option;
        cout << ">> ";

        cin >> option;

         //Following "if" statement is to avoid problem with cin
        //if a non-numeric character is read by cin when
        //it is actually supposed to read an integer.
        if ( !cin )  {
```

```
            cin.clear();
            cin.ignore( INT_MAX, '\n' );
            cout << "Non-numeric option! Try again." << endl;
            continue;
        }

        if (1 == option) {
            string str1 = enterDNAString();
            cout << str1 << " (" << str1.size() << " chars)";
            cout << endl;
        }
        else if (2 == option) {
            string str1 = enterDNAString();
            cout << str1 << " (" << str1.size() << " chars)";
            bool b = checkValidity(str1);
            if(b == true)
                    cout << " : VALID DNA";
            else
                    cout << " : INVALID DNA";
            cout << endl;
        }
        else if (3 == option) {
            string str1 = enterDNAString();
            cout << str1 << " (" << str1.size() << " chars)";
            bool b = checkValidity(str1);
            if(b == true)
                 cout << " : VALID DNA";
            else
                 cout << " : INVALID DNA";
            compliment(str1);
            cout << endl;
        }
         else if (4 == option) {
             return 0;
         }
        else {
            cout << "Integer option, but incorrect! Try again." << endl;
        }
    }
    return 0;
} //end of main
```