

# CSC3002: Introduction to Computer Science

## Assignment 1

### Assignment description:

You should write your code for each question according to the problem requirement. And please pack your **whole project files into a single .zip file**, name it using your **student ID** (e.g. if your student ID is 123456, then the file should be named as 123456.zip), and then submit the .zip file via BB system.

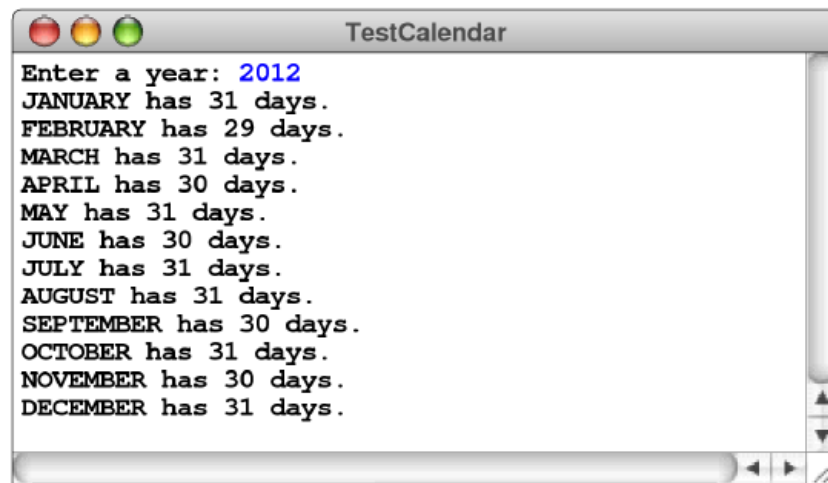
Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we may check whether your program is **too similar** to your fellow students' code using BB.

**Please refer to the BB system for the assignment deadline.** For each day of late submission, you will obtain late penalty in the assignment marks. If you submit more than three days later than the deadline, you will receive zero in this assignment.

**Detailed description on assignment requirement is stated in the following pages.**

## Question 1:

Using the `direction.h` interface as an example, design and implement a `calendar.h` interface that exports the `Month` type from Chapter 1, along with the functions `daysInMonth` and `isLeapYear`, which also appear in that chapter. Your interface should also export a `monthToString` function that returns the constant name for a value of type `Month`. Test your implementation by writing a main program that asks the user to enter a year and then writes out the number of days in each month of that year, as in the following sample run:



```
TestCalendar
Enter a year: 2012
JANUARY has 31 days.
FEBRUARY has 29 days.
MARCH has 31 days.
APRIL has 30 days.
MAY has 31 days.
JUNE has 30 days.
JULY has 31 days.
AUGUST has 31 days.
SEPTEMBER has 30 days.
OCTOBER has 31 days.
NOVEMBER has 30 days.
DECEMBER has 31 days.
```

## Question 2

The genetic code for all living organisms is carried in its DNA—a molecule with the remarkable capacity to replicate its own structure. The DNA molecule itself consists of a long strand of chemical bases wound together with a similar strand in a double helix. DNA's ability to replicate comes from the fact that its four constituent bases—adenosine, cytosine, guanine, and thymine—combine with each other only in the following ways:

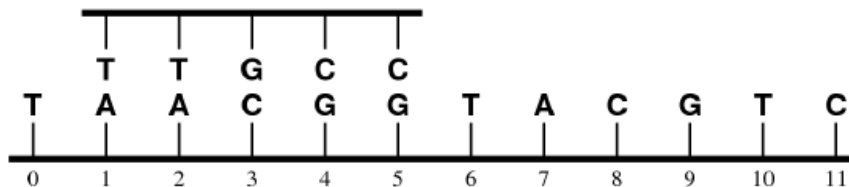
- Cytosine on one strand links only with guanine on the other, and vice versa.
- Adenosine links only with thymine, and vice versa.

Biologists abbreviate the names of the bases by writing only the initial letter: **A**, **C**, **G**, or **T**.

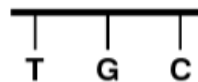
Inside the cell, a DNA strand acts as a template to which other DNA strands can attach themselves. As an example, suppose that you have the following DNA strand, in which the position of each base has been numbered as it would be in a C++ string:



the rules for DNA dictate that this strand can bind to the longer one only at position 1:



By contrast, the strand



matches at either position 2 or position 7.

Write a function

```
int findDNAMatch(string s1, string s2, int start = 0);
```

that returns the first position at which the DNA strand **s1** can attach to the strand **s2**. As in the **find** method for the **string** class, the optional **start** parameter indicates the index position at which the search should start. If there is no match, **findDNAMatch** should return **-1**.

### Question 3

Even though comments are essential for human readers, the compiler simply ignores them. If you are writing a compiler, you therefore need to be able to recognize and eliminate comments that occur in a source file.

Write a function

```
void removeComments(istream & is, ostream & os);
```

that copies characters from the input stream **is** to the output stream **os**, except for characters that appear inside C++ comments. Your implementation should recognize both comment conventions:

- Any text beginning with **/\*** and ending with **\*/**, possibly many lines later.
- Any text beginning with **//** and extending through the end of the line.

The real C++ compiler needs to check to make sure that these characters are not contained inside quoted strings, but you should feel free to ignore that detail. The problem is tricky enough as it stands.

## Question Requirement

**Q1:**

You should write two files: **q1.h** and **q1.cpp**.

Of them, **q1.h** includes the declaration and comments of functions shown below. You need to put comments for each function prototype.

```
enum Month {
    JANUARY, FEBRUARY, MARCH, APRIL, MAY, JUNE, JULY, AUGUST, SEPTEMBER, OCTOBER, NOVEMBER, DECEMBER
};

/*
 * Please add the function description
 */
int daysInMonth(Month month, int year);

/*
 * Please add the function description
 */
bool isLeapYear(int year);

/*
 * Please add the function description
 */
std::string monthToString(Month month);

/*
 * Please add the function description
 */
void q1();
```

**q1.cpp** should include implementations and detailed comments for functions declared in **q1.h**. The test method `void q1()`, should include some test codes to examine the correctness of these functions. The sample output should be:

---

```
Enter a year: 2012
JANUARY has 31 days.
FEBRUARY has 29 days.
MARCH has 31 days.
APRIL has 30 days.
MAY has 31 days.
JUNE has 30 days.
JULY has 31 days.
AUGUST has 31 days.
SEPTEMBER has 30 days.
OCTOBER has 31 days.
NOVEMBER has 30 days.
DECEMBER has 31 days.
```

**Q2:**

You should write two files: **q2.h** and **q2.cpp**.

Of them, **q2.h** should **at least** include the declaration and comments of functions shown below.

```
// Function prototypes
/*
 * Please add the function description
 */
int findDNAMatch(std::string s1, std::string s2, int start = 0);

/*
 *Please add the function description
 */
void q2();
```

**q2.cpp** should include implementations and detailed comments for functions declared in **q2.h**. Of them, **q2()** is the test method which contains the test program.

The test method `void q2()`, should get **user input** from console to specify the **DNA strand to be attached to and a shorter one**. Then it output **all the positions** after matching.

**A sample output should be:**

**\$Enter a longer DNA strand: TAACGGTACGTC**

**\$Enter a shorter one: TGC**

**\$The matching positions should be: 2, 7**

**Q3:**

You should write two files: **q3.h** and **q3.cpp**.

Of them, **q3.h** should at least includes the declaration and comments of functions shown below.

```
// Function prototypes
/*
 * Please add the function description
 */
std::string readFileFromPath(std::ifstream &, std::string f_path);

/*
 * Please add the function description
 */
void removeComments(std::istream & is, std::ostream & os);

/*
 *Please add the function description
 */
void q3();
```

**q3.cpp** should include implementations and detailed comments for functions declared in **q3.h**. Specifically, you should add codes to the test method **q3()** which can get **user input** from console to specify the **absolute file path**. Then it reads the data from the file and output to the new file created under the same directory after removing comments.

The sample output should be:

\$Enter the absolute input file path: F:\Work\input\_file.txt

\$The output file is F:\Work\input\_new.txt

And one sample input file:

```

/*
 * This program generates a list of the powers of two up to an exponent limit
 * entered by the user.
 */
#include <iostream>
using namespace std;

/* Function prototypes */
int raiseToPower(int n, int k);

/* Main program */
int main() {
    int limit; // define the limit of the powers
    cout << "This program lists powers of two." << endl;
    cout << "Enter exponent limit: ";
    cin >> limit;
    for (int i = 0; i <= limit / 2; i++) { // for each power, we print it out
        cout << "2 to the " << i << " = " << raiseToPower(2, i) << endl;
    }
    return 0;
}

/*
 * Function: raiseToPower
 * Usage: int p = raiseToPower(n, k);
 * -----
 * Returns the integer n raised to the kth power.
 */
int raiseToPower(int n, int k) {
    int result = 1;
    for (int i = 0; i < k; i++) {
        //i = i + k;
        result *= n;
    }
    /* return result;
}

```

## The sample output file will be

```

#include <iostream>
using namespace std;

int raiseToPower(int n, int k);

int main() {
    int limit;
    cout << "This program lists powers of two." << endl;
    cout << "Enter exponent limit: ";
    cin >> limit;
    for (int i = 0; i <= limit / 2; i++) {
        cout << "2 to the " << i << " = " << raiseToPower(2, i) << endl;
    }
    return 0;
}

int raiseToPower(int n, int k) {
    int result = 1;
    for (int i = 0; i < k; i++) {
        result *= n;
    }
}

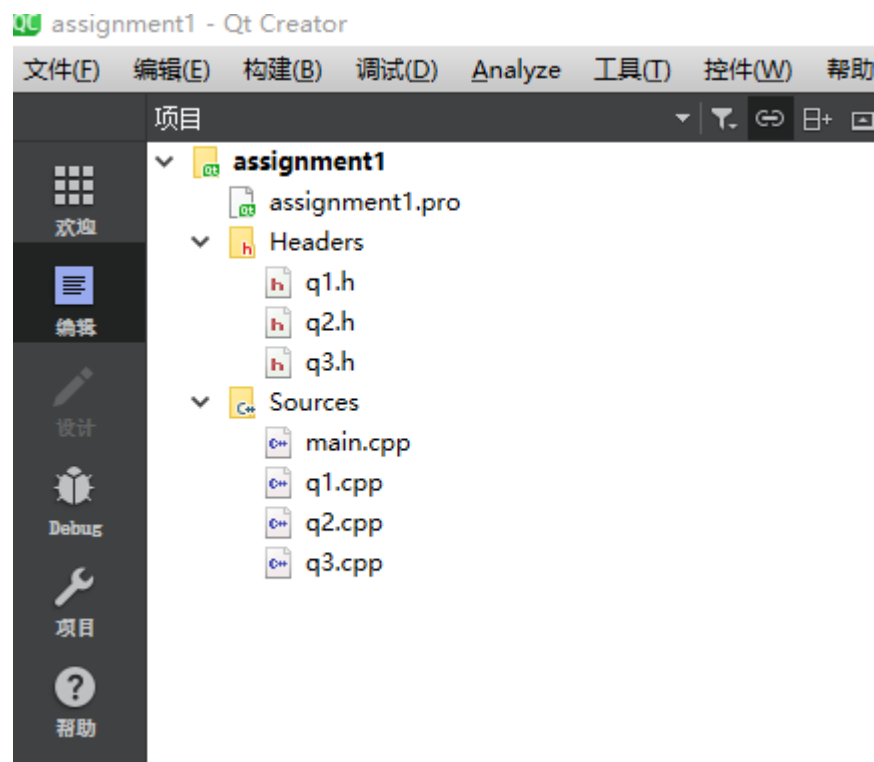
```

**In the output file, you code should be able to deal with the case with only “/\*” (the corresponding “\*/” does not exist) in which all characters following “/\*” will be removed.**



## Assignment Submission

Please find the attached assignment1.zip in the BB system which contains an QT project created for assignment1 and you can edit the corresponding files for these questions according to the requirements mentioned above.



Note that the main.cpp is already created.

```
int main() {  
    q1();  
    q2();  
    q3();  
    return 0;  
}
```

After you test all your questions, **zip the whole project in one file named XXX.zip (XXX is your student ID and your name is not required)** and then submit it to the BlackBorad (BB) system. **Please pay attention to this because only source files cannot be executed on our computer which will affect your scores!!!!!!!!!!!!!!**

## **Marking scheme**

For each question:

- **20%** Marks will be given to students who have submitted the program **on time**.
- **20%** Marks will be given to students who wrote the program that meet all the **requirements** of the questions
- **20%** Marks will be given to students who programs that can be compiled without **errors and warnings**
- **20%** Marks will be given to students whose programs produce the **correct output** if their programs can be compiled.
- **20%** Marks will be given to students who demonstrate good **programming habit and style (e.g. comments and code layout)**.

## Q&A

1. How to check whether the assignment is submitted on time?

We directly check it through BB system

2. How to check whether the assignment can be compiled?

Note that we **strongly suggest** you to use the QT IDE since your code tested on other IDE may not work on QT. If you want to use other IDE, please test your code again on QT to avoid such errors.

If the QT IDE generates any **errors** when building your project, 20% mark will be missed for each question. If **any warnings** are produced but your code can run, 20% mark will be missed for all 3 questions.

3. How to check the assignment gives correct output?

We will run your code if it can be compiled (warnings are tolerated). You **test code** should be correct and **output** is also correct. User-friendly test code and output is preferred.

4. How to check the programming style?

- **Comments.**

You should include three kinds of comments in your code:

- a. File comments in the beginning of each code file, for example,

```
/*
 * File: AddIntegerList.cpp
 * -----
 * This program adds a list of integers. The end of the
 * input is indicated by entering a sentinel value, which
 * is defined by setting the value of the constant SENTINEL.
 */

#include <iostream>
using namespace std;
```

- b. Function comments just before each function implementation. For example,

```
/*
 * Function: raiseToPower
 * Usage: int p = raiseToPower(n, k);
 * -----
 * Returns the integer n raised to the kth power.
 */

int raiseToPower(int n, int k) {
    int result = 1;
    for (int i = 0; i < k; i++) {
        result *= n;
    }
    return result;
}
```

- c. Statement comments on the right side of some statements **if necessary**. You do not have to write comments for each statement.

- **Code layout.**

**Beautiful !!!!**

The following points should be pay attention to:

- a. The indentation used**
- b. {} used for function, for structure, while structure, switch structure.**
- c. meaningful variable name.**

**We strongly suggest you to adhere to the coding style used in our textbook, which will be followed by us to check your code.**