Files and Functions

Object Oriented Programming - Programming Coursework #5 (16 pts)

Due date : June 11, 2020 5pm

Please note the new source code and execution requirements (Secs. 3.1 and 3.2).

1 Task

The objective of this coursework is to have a hand-on experience on file and function processing. The task is to implement two programs. The first program (Program1) will convert an integer to a binary string (8 pts): The program will read a positive integer in base 10 from an input data file ('Program1Input.txt'), and generate an output file ('Program1Output.txt') that contains a string representing the same number in binary (in base 2). The second porgram (Program2) will take a non-negative integer, say n and return the n-th number of the Fibonacci sequence (8 pts): Here we say that the 0-th and 1st entries of the Fibonacci sequence are 0 and 1. The n-th number of the Fibonacci sequence for $n \geq 2$ is then defined as the sum of the $(n-1)$-th and $(n-2)$-th numbers.

2 Program instructions and specifications

All submitted codes have to satisfy the **source code requirements** and **the execution requirements** described in Secs. 3.1 and 3.2, respectively, as well as the compilation requirements (Sec. 3.3): If the codes do not meet these requirements, the students will receive 0 point.

2.1 Program1

Input file: 'Program1Input.txt' to be provided by the CW assessor. This contains a positive integer.

Output file: 'Program1Output.txt' to be generated by Program1. This should contain a sequence of zeros and ones corresponding to the binary representation of the integer value provided in 'Program1Input.txt'.

2.2 Program2

Input file: 'Program2Input.txt' to be provided by the CW assessor. This contains a positive integer.

Output file: 'Program2Output.txt' to be generated by Program2. Assuming that the input integer value (provided in 'Program2Input.txt') is n, Program2Output.txt file should contain an integer corresponding to the n-th number of the Fibonacci sequence.

Sample inputs and outputs.

'Program1Input.txt' file:

```
259
```

'Program1Output.txt' file:

```
100000011
```

‘Program2Input.txt’ file:

```
30
```

‘Program2Output.txt’ file:

```
832040
```

‘Program2Input.txt’ file:

```
6
```

‘Program2Output.txt’ file:

```
8
```

2.3 Assessment

Program1 will be assessed based on five different instances of ‘Program1Input.txt’. ‘Program1Input.txt’ file will be located at the same directory as the executable file of Program1. When executed, with an instance of ‘Program1Input.txt’, Program1 will generate ‘Program1Output.txt’ file in the same directory. Similarly, Program2 will be assessed based on five instances of ‘Program2Input.txt’ (in the same directory as the executable file of Program2) and it should generate ‘Program2Output.txt’ file.

For this CW, your programs do not have to handle exceptions caused by incorrect input data types. For instance, for Program1, ‘Program1Input.txt’ won’t contain anything else then a positive integer.

3 What to hand in for assessment

Please submit the source codes for Program1 and Program2, and a brief report (a single report for both programs) via Black Board. Comments in the codes and the report should be written in English. The report should provide code comments explaining the algorithms. If the submission does not include a report, the maximum possible points will be kept at 40% of the full points. Please format the submission as a single zip file ‘StudentID_Name.zip’ that includes the files of the source codes and the report,

e.g.,
20201111_KwangInKim.zip
- Program1.cpp
- Program2.cpp
- Report.pdf

The submitted source codes should contain ‘Program1.cpp’ and ‘Program2.cpp’ files each including the id and name of the submitting student.

In “Program1.cpp”

```
//StudentId Name
#include <iostream>
Your code here...
```

3.1 Source code requirements

The submitted Program1 code should implement and call a function ‘myIntegerToBinary’ that receives an input integer, say ‘n’, and returns a string containing the binary representation of n. myIntegerToBinary should not call any function other than itself, and two basic string functions ‘to_string’ and ‘reverse’.[1]

[1]It is possible to make myIntegerToBinary function without having to use ‘to_string’ or ‘reverse’ functions.

```
string myIntegerToBinary(int n)
{
    %Your code here ....
}
```

The Program2 code should implement and call a function ‘myFibonacci’ that receives an input integer, say ‘n’, and returns an integer corresponding to the n-th number of the Fibonacci sequence. myFibonacci function should not simply memorize the Fibonacci sequence and it should actually calculate the n-th number of the Fibonacci sequence from scratch. Please note that we define the 0-th and 1st numbers of the Fibonacci sequence as 0 and 1, respectively. myFibonacci should not call any function other than itself.

```
int myFibonacci(int n)
{
    %Your code here ....
}
```

The submitted codes should implement the required operations from scratch: **Calling library functions that perform such operations from any part of the submitted codes is not allowed**.

3.2 Execution requirements

For assessing Program1, we will select the number in each instance of ‘Program1Input.txt’ from $\{1, \ldots, 100\}$. Similarly for Program2, the number in each instance of ‘Program2Input.txt’ will be selected from $\{1, \ldots, 30\}$. In this case, both Program1 and Program 2 should be terminated within a second at UNI06 server.

3.3 Compilation requirements and instruction

Each submitted code needs to be compilable by the GNU C++ compiler and in the Linux environment. Specifically, they should be **compilable and executable in UNI06 server**. Students can verify the correct compilation and execution of their codes by following the instructions provided in ‘howtocompile.pdf’ document accompanying the CW1 announcement.