COMP 1602 – Computer Programming II

2023/2024 Semester 2

Lab #10

Binary Files

1. The file `Lab10.cpp` contains code which writes three integer values to a binary file `Lab10.bin`. The file is then reopened, and the three integer values are read and displayed on the monitor.

(a) In `Lab10-Question1.cpp`, write code in *main* to store the integers 1 to 100 in the binary file, `Lab10-Question1.bin`. Close the file when you are done.

(b) In `Lab10-Question1.cpp`, write code in *main* to read the 100 integers from the binary file created in (a). Verify that the values are 1 to 100. Close the file when you are done.

(c) Assume that you do not know how many integers are stored in the binary file. Write code in *main* to read all the integers from the file. Display the values read on the monitor.

2. The file, `Lab10-Question2.cpp` contains the declaration of the following struct:

```
struct Student {
   char name [25];
   int mark;
   char grade;
   bool passed;
};
```

(a) Write a function, *readStudents*, which reads the *name* and *mark* of several students from a text file, the name of which is passed as a parameter. The function stores the data in an array of *Student* structs and returns the number of structs stored in the array. The function has the following heading:

```
int readStudents (Student students[], char filename[])
```

NB: Data for the *grade* and *passed* fields are NOT present in the text file. Also, assume that the last line of data in the text file contains the word, "END", only.

(b) Write a function, *saveStudents*, which stores the array of *Student* structs in a binary file, the name of which is passed as a parameter. When each *Student* struct is being written to the file, data for the *grade* and *passed* fields are determined based on the following ranges:

Grade	**Passed**
A: 80 – 100 B: 65 – 79 C: 50 – 64 F: 0 – 49	≥ 50: True < 50: False

The heading of the function is as follows:

```
void saveStudents (Student students[], int numStudents,
                   char filename[])
```

(c) In *main*, call *readStudents* with the file, `marks.txt`.

(d) In *main*, call *saveStudents*. The name of the binary file to save the *Student* structs is `Lab10-Question2.bin`.

(e) In *main*, read all the *Student* structs from the binary file. Call the function, *displayStudent* (already written) to display the data on each student read from the file. Verify that the data has been correctly read from the file and that the data for the *grade* and *passed* fields were correctly determined.

3. (Copy your solution for `Lab10-Question2.cpp` to `Lab10-Question3.cpp`).

The *Student* struct from Question 2 requires 36 bytes of storage (*sizeof* can be used to find the number of bytes of storage). The first 25 bytes of each struct is used for the *name* field. The remaining 11 bytes is used for the *mark*, *grade*, and *passed* fields.

(a) Write a function in `Lab10-Question3.cpp` to search the binary file containing the *Student* structs from Question 2(d) for a particular student's name. The heading of the function is as follows:

```
int findStudent (char filename[], char studentName[])
```

The approach to use is to check the first 25 bytes of the file to determine if it is the same as the name being searched for. If so, return *true*. Otherwise, advance the *get* pointer 11 bytes, and search again. Repeat this process until end-of-file is reached or the name is found.

(b) Write a *main* function in `Lab10-Question3.cpp` which requests the user to input a student's name and then searches the binary file to determine if the student is present.

4. The file, `Lab10-Question4.bin`, contains a secret message which is stored in 40 ASCII characters. Write code to find and display the message.

The first character of the message is found after skipping 17 integers. To find the second character of the message, you must skip over one integer. To find the third character of the message, you must skip over two integers. To find the fourth character of the message, you must skip over three integers. Follow this pattern until you come to the 40th character of the message.

Hint: You need to use a nested *for* loop to solve this problem. The inner *for* loop will skip over the required number of integers in the binary file.

End of Lab #10