Lab 12

CSE 165: Object Oriented Programming

Spring 2022

(100 points)

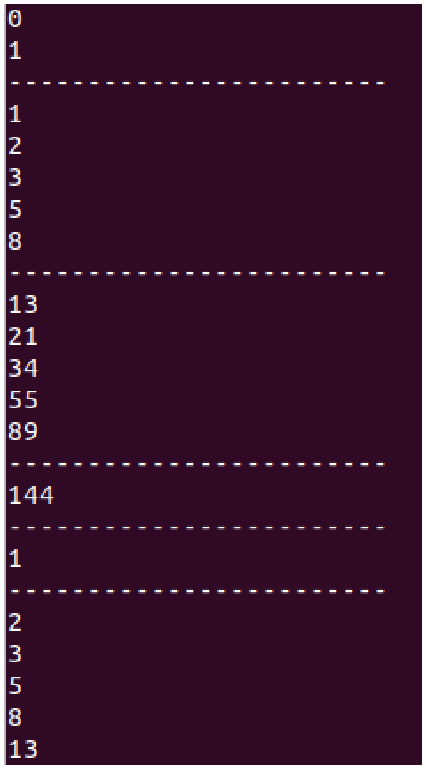
This programming assignment has five tasks, complete each task as instructed. Write a separate file for each of the following tasks. To submit your assignment, please organize your code in the folder ”Lab12” by placing your code in its corresponding sub-folder. For example, store your code for task 1 in the following directory ”Lab12/1/”. Then, submit the compressed version of folder Lab12 to CatCourses. Submissions must arrive by one minute before the lab section of week 15 (4/25 – 4/29). All of the files you need for this programming assignment are available in a ZIP archive file called “Lab12.zip”.

1. **Fibonacci Sequences (15 Points)**

   Create a class called `Fib` that returns the next value in a Fibonacci sequence every time you call its `next()` method. The class should only have two static members called `last` and `secondLast` to store last two numbers in Fib object. The `next()` method should have an argument that is a bool with a default value of false such that when you give the argument with true it ”resets” the function to the beginning of the Fibonacci sequence.

   Submit your code in a file named `Fib.h`. Test your code with `fib.cpp`.

   Expected Output:



```
0
1
------------------------
1
2
3
5
8
------------------------
13
21
34
55
89
------------------------
144
------------------------
1
------------------------
2
3
5
8
13
```

2. **Incident Monitor (20 Points)**

   Create a class called `Monitor` that keeps track of the number of times that its `incident()` method has been called. Add a `print()` method that displays the number of incidents.

   Save your class in a file named `Monitor.h`. Test your code with the `monitors.cpp` file.

3. **Namespaces (15 Points)**

   Use the files `Functions.h` and `namespaces.cpp`. There is a namespace called `MyLib` defined in `Functions.h`. Extend that namespace in a file called `MoreFunctions.h` so that the code compiles and runs without errors and matches the expected output. Submit your `MoreFunctions.h` file.

   Expected Output:

   ```
   f
   someFunction
   ```

4. **Vector Operators (20 Points)**

   Use the files `Vec.h` and `vecs.cpp`. Extend the `Vec` class defined in `Vec.h` in such a way that the `vectors.cpp` file compiles and works correctly. Submit your modified version of `Vec.h`.

   Expected Output:

   ```
   (9, 10)
   ```

5. **Matrix Operators (30 Points)**

   Write a `Mat` class that has four float members, which are the four elements of a 2x2 matrix. Implement all constructors, methods, and operators so that the file `matrices.cpp` compiles and works correctly. You may use the `Vec.h` file from the previous exercise.

   Expected Output:

   ```
   (3, 14)
   ```