CS 247 Spring 2014
Assignment 2 Deliverables

Assignment 2 deliverables are worth 5% of your overall course mark.
No groupwork is permitted on this assignment.

Due dates

The questions are due on **Monday June 9, 2014** at **12:00 noon.** They are to be submitted electronically through **Marmoset**.

Marking Rubric

Programming questions will be marked for design quality and programming style as well as correctness. See the marking rubric for details on how your programs' design and programming style will be marked.

Q1 [70 marks] PImpl Idiom

You are to implement an *immutable* Date ADT whose data representation is hidden within a nested structure (called the 'private implementation' idiom, or 'pimpl').

Submission

Submit your answer to Marmoset in a zip archive named **a2q1.zip**. It should include

- `Date.h`
- `Date.cpp`

Make sure that your code runs with our provided DateTestHarness on the undergraduate environment (e.g., that your programming environment does not implicitly include libraries for you).

Marking

The test scripts and marking scheme for this question are structured according to the following increments:

1. create Date objects from valid Date input: constructor, operator<<
2. accessors: day(), month(), year()
3. comparison operators: operator==, operator!=, operator<, operator<=, operator>, operator>=
4. arithmetic operations: incDays(), incMonths(), incYears()
5. copy constructor, assignment
6. static function today() (hint: take a look at the C library ctime)
7. operator>>
8. recover from invalid Date values (i.e., throw exception or round down to a valid value, as specified in the specifications)

We strongly recommend that you implement each increment to completion before progressing to the next increment.

The output of your program with be checked automatically using Marmoset — it must match exactly the output of our solution. We provide an executable `Dates` that runs on the Linux machines in the undergraduate environment. You can use this executable and the UNIX command `diff` to compare the output of your program against our solution.

Q2 [80 marks] Exceptions, Smart Pointers, Modules

You are to implement the UserAccount ADT, as well as design and implement the Userid and Password ADTs, and provide a makefile that automates incremental compilation.

Submission

Submit your answer to Marmoset in a zip archive named **a2q2.zip**. It should include

- `UserAccount.h, Password.h, Userid.h`
- `UserAccount.cpp, Password.cpp, Userid.cpp`
- `Makefile`

Make sure that your code runs with our provided TestHarness on the undergraduate environment (e.g., that your programming environment does not implicitly include libraries for you).

Marking

The test scripts and marking scheme for this question are structured according to the following increments:

1. construct valid UserAccount objects
2. deactivate accounts, check whether accounts are deactivated
3. reactivate accounts
4. delete accounts
5. authenticate accounts
6. handle invalid userids
7. handle invalid passwords

We strongly recommend that you implement each increment to completion before progressing to the next increment.

The output of your program with be checked automatically using Marmoset — it must match exactly the output of our solution. We provide an executable `Passwords` that runs on the Linux machines in the undergraduate environment. You can use this executable and the UNIX command `diff` to compare the output of your program against our solution.

Q3 [30 marks] ADT Design

You are to explain some of the ADT design decisions you made in your answer to question Q2.

Submission

Put your answers in a text file titled `ADT.txt`, and submit the file electronically to Marmoset.