UNIVERSITY OF WATERLOO

CS247

Software Abstraction and Specification
Midterm Examination

Spring 2011

Date: 23-June-2011
Time: 4:30–6:30 PM
Permitted Aids: Supplemental Sheet only

Signature ____

Instructions: (Read carefully before the exam begins):

1. Before you begin, make certain that you have one Exam Booklet with pages numbered 1–16 printed double-sided.
2. This is a **closed-book exam.** The only permitted aid is the provided *Supplemental Sheet*.
3. There are a total of 7 questions, for a total of 90 marks.
4. The marks assigned to each question are shown at the beginning of the question; use this information to organize your time effectively.
5. Place all your answers in the spaces provided on these pages. There are 2 extra sheets at the end of the exam. **Do not write answers on the Supplemental Sheet.** It will not be marked.
6. You do not need to write comments in your code if you write self-commenting code.
7. Questions will not be interpreted. Proctors will only confirm or deny errors in the questions. If you consider the wording of a question to be ambiguous, state your assumptions clearly and proceed to answer the question to the best of your ability. You may not trivialize the problem in your assumptions.
8. Cheating is an academic offense. Your signature on this exam indicates that you understand and agree to the University's policies regarding cheating on exams.

Question	out of	mark	marker's initials
1	15		
2	5		
3	8		
4	10		
5	9		
6	9		
7	7		
8	20		
Total	83		

Question 1 [15 marks] - ADT Design

Design an ADT for valid values of Licence Plates for cars and trucks in Ontario. A valid value is a sequence of two to eight characters, where a "character" is a capital letter, a digit, or a space. (In reality, some sequences are invalid because they are objectionable, but we will ignore this detail.) In addition, Licence Plate values must be **unique**.

Your ADT should be useful in any program that maintains information about licence plates or any program that identifies or searches for vehicles by their licence plate.

Your answer to this question is in several parts. Read all of the parts of the question before starting your answer.

a) Provide a complete class *declaration* of your ADT, including nonmember functions, `const`, and default arguments.

b) If you are implementing your own default constructor, state your choice of default initial value and explain why it is appropriate. If you are not implementing your own default constructor, explain why. In either case, restrict your answer to 1 sentence.

c) State whether objects of your ADT are mutable or immutable, and explain (briefly) the rationale of your decision.

d) Explain briefly your rationale for making the member functions in your ADT member functions, and making nonmember functions in your ADT nonmember functions.

BONUS [2 marks] Explain how your ADT ensures that each constructed Licence Plate object has a unique value.

Question 1 [15 marks] - ADT Design (cont.)

Questions 2-6 make use of information on the Supplemental Sheet

On the *Supplemental Sheet*, there is a UML model and C++ class declaration for a composite object WatCard. The Supplemental Sheet also contains class declarations for Date and Money ADTs that are used by WatCard objects and components.

Questions 2 through 6 ask you to implement code fragments of the composite object WatCard. Your answers to these questions should agree with one another (e.g., they should compile, link, and work together).

For full marks, encapsulate data and avoid redundant code.

Question 2 [5 marks] - Composite Object Initialization

Provide the C++ *definition* for the constructor of the WatCard class. The constructor should initialize all data members and should produce an object that adheres to the provided UML model:

- it should have a unique barcode value
- it should have no PIN
- it should be inactive
- it should have an expiry date of 2 years from today
- it should have a Flex Account with a balance of $0.00
- it should have no other accounts

The initialization of object members should be as efficient as possible.

Question 3 [8 marks] - Essential Methods

a) Consider the following methods for the WatCard class: default constructor, destructor, copy constructor, assignment operator=. For each, state whether the compiler-generated version (if it exists) would be appropriate; explain your answer. (Credit is awarded only for correct explanations; not for correct determinations.)

b) Provide the C++ *definition* of the assignment operator= for the Meal Plan Funds class (regardless of whether it makes sense to implement this operator).

Question 4 [10 marks] - Abstract Base Class

a) Provide the C++ *declaration* for the Account class.

- Your declaration should conform to the UML model
- Define Account to be an *abstract base class*.
- Add an accessor and mutator (not shown in the UML model) that allow derived classes to access and set the value of data member balance. These methods should not change the *public* interface of Account (as modelled in UML). You **may** use these methods in your answers.
- Use const appropriately.

b) Provide a C++ default implementation for the method Account::purchase(). Assume no discounts and no HST. If there are sufficient funds in the account, the method subtracts the amount (given by parameter) from the account's balance and returns *true*. If there are insufficient funds in the account, then no funds are subtracted and the method returns *false*.

Question 5 [9 marks] - Derived Class

a) Provide the C++ *definition* for the constructor for class Meal Plan Funds. (Think about whether a Meal Plan Funds object should refer to a *shared* Meal Plan object or should refer to its *own copy* of a Meal Plan object.)

b) Provide a C++ definition for the method MealPlanFunds::purchase() that applies the meal plan's discount to the purchase. Assume that a discount is a whole number (e.g., value 50 represents a 50% discount).

c) Provide a C++ definition for the method MealPlanFunds::endMealPlan(toAccount) that removes the Meal Plan and transfers any unused funds to the given account.

Question 6 [9 marks] - Polymorphic Code

Provide the C++ *definitions* for the following WatCard methods.

a) WatCard::purchase() - If the purchase is made to a Meal Plan Funds account or a Transfer Meal Plan Funds account and there are insufficient funds, then instead charge the purchase to the Flex Dollars account.

b) WatCard::addDollars() - Add funds to the Flex Dollars account.

c) WatCard::transactionHistory() - For each of the WatCard's accounts, stream all transactions that were conducted between the given start and end dates, inclusive.

Question 6 [9 marks] - Polymorphic Code (cont.)

Question 7 [7 marks] - Object-Oriented Design

a) State whether a Transfer Meal Plan Funds object is substitutable for a Meal Plan Funds object (i.e., whether Transfer Meal Plan Funds should be a subclass of Meal Plan Funds). Use the rules of the *Liskov Substitutability Principle* to defend your answer. (Credit is awarded only for justification, not for correctly determining substitutability.)

b) Describe the "Law" of Demeter. Based on what you know about the UML model and its implementation, explain why you think the WatCard composite class adheres to or violates this design principle.

Question 8 [20 marks]

a) [1 mark] Consider a function declaration:

```
void func (Object &o);
```

If the function does not modify the argument `o`, what is the disadvantage of *not* declaring `o` to be constant?

b) [1 mark] Explain why `operator<<` and `operator>>` are *not* be member functions.

c) [5 marks] Rewrite the following class declaration so that the *implementation is hidden*. Your answer should include the declaration of the hidden implementation.

```
class Hour {
    int hour_;
public:
    Hour (int);
    int hour() const;
};
```

Question 8 [20 marks] (cont.)

d) [2 marks] Write an assertion to be included in the constructor for Hour (from question 7c) that (1) checks that the object's value is between 0 and 23, inclusive and (2) issues a descriptive error message whenever the assertion is violated.

e) [2 marks] Give one reason why it would be better to use an assertion that terminates the program if a client-provided value for an Hour object is illegal, rather than converting the illegal value into a legal value and carrying on.

f) [2 marks] Amend the following declarations to break the dependency cycle

```
#ifndef A_H
#define A_H

#include "B.h"

class A{
   B *b;
};

#endif
```

```
#ifndef B_H
#define B_H

#include "A.h"

class B {
   A a;
};

#endif
```

g) [2 marks] Write a *makefile rule* that (1) compiles the source code file (`B.cpp`) for the class `B` from question 7f, to produce an object file (`B.o`), but (2) compiles the file only if none of the files it depends on has changed since the last time the file was compiled. Assume only the provided declarations above (not your amended declarations).

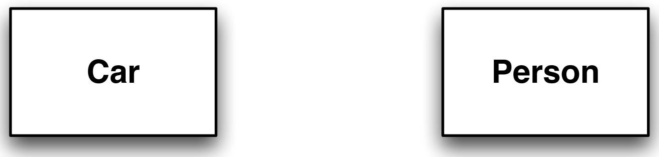
Question 8 [20 marks] (cont.)

g) [1 mark] What is a self-testing object?

h) [4 marks] Annotate the following UML model with associations, role names, and multiplicities, so that the model specifies that

i. a car has exactly 1 owner

ii. a car has some number of drivers



Extra Sheet #1

Extra Sheet #2