**CS 247**  
Software Engineering Principles  
Midterm Examination **Solutions**  
Spring 2017

| Last Name: | ___________________________ |
| First Name: | ___________________________ |
| ID # | ___________________________ |

**Date:** Thursday, June 22, 2017  
**Time:** 4:30 – 6:20 pm  
**Instructor:** C. Kierstead

**Lecture Section:** 001  
**Exam Type:** Closed book  
**Additional Materials Allowed:** None

---

**Instructions:**  
(Read carefully before the exam begins)

1. Before you begin, make certain that you have one Exam Booklet with 14 pages, an OMR card, and a reference sheet.
2. All solutions must be placed in this booklet except for the answers to Part A. Do not detach the pages of this booklet. Complete the OMR card in HB or 2HB pencil.
3. Use the amount of space provided as a guide to the amount you should be writing. If you need more space to complete an answer, you are likely writing too much. However, if you do need more space, use the last page, and indicate that you have done so in the original question.
4. You do not need to write comments in your code if you write self-commenting code.
5. All C++ questions must be answered using C++11 or C++14 (no generics, or lambdas unless specified otherwise in the question).
6. Questions will not be interpreted. Proctors will only confirm or deny errors in the question. 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.
7. 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.
8. Relax! Read this instruction as often as needed.

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks Given</th>
<th>Out Of</th>
<th>Marker’s Initials</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part A: 1-24</td>
<td>32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part B: 1-8</td>
<td>13</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part C: 3</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part C: 4</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Part C: 5</td>
<td>19</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>79</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Student signature**
Part A: [32 marks] Multiple Choice/True-False

Fill in your answer on the OMR card with your exam, not (just) on the exam paper itself. Having your answer in both is okay, but we’ll be scanning the card to mark it automatically, so your answer must be present on the card. Your answer must be in HB or 2HB pencil so that it can be scanned.

The multiple-choice questions may have more than one correct answer. If that is the case, you must select all valid answers to receive the full mark. There is no penalty for guessing incorrectly i.e. your mark for the question will not go below 0.

1 [1 mark] Outside-in Design starts with developing the:
   a) ADT interface design  
   b) ADT implementation  
   c) client use scenarios

2 [2 marks] What are valid options for a constructor in C++ to report an invalid value?
   a) output an error message  
   b) set the data members to a different, legal value  
   c) exit the program via the exit() system call  
   d) raise an exception  
   e) return an error code

3 [2 marks] Given the following code, what is its output?
   ```cpp
   #include <iostream>
   class A {
   int i;
   public:
   A() : i{0} {}  
   void foo() const { std::cout << i << std::endl; }
   void foo() { i++; }  
   int get() const { return i; }
   }
   int main() {
   A a;
   a.foo();  
   std::cout << a.get() << std::endl;
   }
   ```
   a) code doesn’t compile since foo is declared twice  
   b) code doesn’t compile since call to a.foo() is ambiguous  
   c) outputs: 0
   d) outputs: 1
   e) code won’t run properly since i wasn’t initialized and it might overflow

4 [2 marks] If a class Foo has the following public constructor declared:
   ```cpp
   Foo( int a = 1, char b = '\t', bool flag = false);
   ```
   assuming that the constructor is correctly implemented, which of the following are valid object definitions?
   a) Foo f1;  
   b) Foo f2(23);  
   c) Foo f3(23, 'a');  
   d) Foo f4(23, 'a', true);  
   e) Foo f5('b');

5 [1 mark] If a class is declared final, it cannot be a base class.
   a) True  
   b) False
Part A: [32 marks] Multiple Choice/True-False (continued)

6 [2 marks] Assume that the ADT Matrix is implemented as a class. Which of the following operations must not be members of the Matrix class?
   a) operator=
   b) operator+
   c) operator<<

7 [1 mark] An attribute-based ADT is primarily used to restrict the range of legal values for an ADT. All of its members must check and enforce the valid range of values.
   a) True
   b) False

8 [1 mark] An entity-based ADT must provide copy operations (constructor and assignment), as well as the ability to compare two objects of the entity type.
   a) True
   b) False

9 [2 marks] The compiler generates which of the following special member functions?
   a) default constructor
   b) destructor
   c) copy and move constructors
   d) copy and move assignment operators
   e) equality operator

10 [1 mark] A using directive should never be placed before an #include directive in an implementation file since it affects the set of names visible in the headers subsequently included.
    a) True
    b) False

11 [2 marks] Given the following code, what is its output?
   ```cpp
   #include <iostream>
   class MyException {};
   class SomeError: public MyException {};
   class OtherError: public SomeError {};
   void foo() { throw OtherError(); }
   int main() {
      try {
         foo();
      } catch( MyException & e ) {
         std::cout << "my exception\n";
      }
      std::cout << "done\n";
   }
   ```
    a) doesn't compile
    b) "my exception\n"
    c) "my exception\ndone\n"
    d) "done\n"

12 [1 mark] Representation invariants can specify either structural invariants of the chosen data structure or value invariants of the chosen data structure/algorithm, but not both.
    a) True
    b) False
Part A: [32 marks] Multiple Choice/True-False (continued)

13 [1 mark] The following diagram is an example of:

| a) class model |
| b) object model |
| c) sequence model |

[Diagram of a sequence diagram with interactions labeled print(), printReceipt(), and print().]

14 [2 marks] The following code fragment is best described by which of the diagrams?

class B { ... };
class A { vector<B> bs; ... };

- a) [Diagram a) showing 1:* relationship]
- b) [Diagram b) showing 1:5 relationship]
- c) [Diagram c) showing 1:0..1 relationship]
- d) [Diagram d) showing 1:1 relationship]
- e) [Diagram e) showing a "diamond" relationship]

15 [2 marks] The following code fragment is best described by which of the diagrams?

class B { ... };
class A { vector<B*> bs; ... };

- a) [Diagram a) showing 1:* relationship]
- b) [Diagram b) showing 1:5 relationship]
- c) [Diagram c) showing 1:0..1 relationship]
- d) [Diagram d) showing 1:1 relationship]
- e) [Diagram e) showing a "diamond" relationship]

16 [1 mark] The following diagram is an example of:

| a) aggregation |
| b) association |
| c) composition |
| d) generalization |
| e) self-association |

[Diagram of an association relationship between A and B with a "diamond" symbol.]
Part A: [32 marks] Multiple Choice/True-False (continued)

Complete each of the following phrases by filling in "bubble code" for the term on the reference sheet that best completes the sentence. For example, if the code associated with "Graph" was "ABC", you'd fill "ABC" as the answer for " ___ is an ADT consisting of a unique set of nodes with undirected edges between any unique pair of nodes."

17 [1 mark] ________ is a user-defined type that reveals the range of values and operations, but not the implementation.
   a Abstract Data Type

18 [1 mark] ________ represents a value. Objects of the same type, and with the same attribute values, are considered to be identical.
   b or c Attribute-based ADT or Value-based ADT

19 [1 mark] An operation is ________ if it leaves the program in a valid state after terminating by throwing an exception.
   a Exception-safe

20 [1 mark] ________ ensures that all basic invariants of all objects are maintained, no resources are leaked, and that the program state is returned to what it was before the call.
   b Strong guarantee

21 [1 mark] ________ is an abstraction of the software system's code. The code may be already developed, or about to be developed.
   a Software model

22 [1 mark] ________ captures a snapshot of the run-time structure of an executing program. It describes class instances and the links between them.
   b Object diagram

23 [1 mark] ________ is the measure of dependency between modules.
   a Coupling

24 [1 mark] ________ is the measure of relatedness within a module.
   e Cohesion
Part B: [13 + 15 + 2 BONUS marks] Short Answer

1. [1 mark BONUS] What does the term RAII stand for?
   Answer must be exactly correct: Resource Acquisition is Initialization or Resource Acquisition is Initialisation

2. [1 mark BONUS] What does the term UML stand for?
   Answer must be exactly correct: Unified Modeling Language or Unified Modelling Language

3. [2 marks] There are three different ways that an ADT can have representation exposure. Describe two of them.
   Any two of:
   - public data members
   - object passed in is incorporated directly into the ADT and client still has a handle on it e.g. pointer
   - ADT returns a pointer/reference/iterator to the internal representation.

4. [3 marks] Why should a destructor never raise an exception?
   Might already be in the process of trying to find a handler, in which case, this destructor is being invoked as the run-time stack is being unwound and don't know which exception has priority in being handled.

5. [3 marks] You have been given the following class declaration for a Scrabble dictionary (assume necessary includes have been made). Make the necessary changes to the declaration i.e. write the code (declarations only) to implement the Pimpl idiom.

```cpp
class Dictionary {
    std::vector<std::string> words;

    public:
        // reads in words from file
        Dictionary( const std::string & filename ) { ... }
        // returns true if word is in dictionary
        bool isWordValid( const std::string & word ) { ... }
};
```

Adds a private class/struct declaration for the implementation e.g. class Impl;
Adds a private pointer of the implementation type e.g. Impl* ptr;
 Strikes out the private vector.
Part B: [13 15 + 2 BONUS marks] Short Answer

6 [1 mark] How are the Facade and Adapter design patterns similar to each other?

Both put a class (interface) between the client and the other class (or classes).

[2 marks] How are the Facade and Adapter design patterns different from each other?

Adapter completely encapsulates the class(es) being adapted. Facade simplifies some action, but subsystem is still available.

or

They differ in intent i.e. Facade simplifies operation(s) while Adapter converts an interface (or interfaces).

Not okay if the answer relies on # of classes since either one can be applied to 1 or more classes.

7 [1 mark] The MVC design pattern is made up of which two design patterns?

Observer and Strategy, though Facade could be argued. Also okay if mention Composite (used in GUI).

Not Adapter, though [-0.5].

8 [1 mark] If we aren't concerned about efficiency, we might want to use the Copy-Swap Idiom when implementing copy assignment. Explain why.

(Provides safety) If deep copy fails, object left in original state. Or mentions that provides a strong exception guarantee.

[-2] Not okay if just describes what copy-swap does.
Part C: Q3: [11 marks] Interface Specifications

Read the whole question before you answer it.

The problem context for this question is that of the Graph class, as implemented in assignment 2. In particular, assume that the Graph is implemented as a singly-linked list, ordered by the User::id(), which returns EmailAddress::id(), which returns a string. The node definition is on the reference sheet.

The specification fields for the Graph that describe its hidden state are: a set s of Users, where each User has associated with it an email address and a set of edges, e. Note that we're using "set" in the mathematical sense i.e. unordered, with no duplicate elements, and potentially empty.

a) [4 marks] Write the representation invariant for the Graph ADT based upon the constraints as defined in assignment 2.

Notation doesn't have to be exact, but should be reasonable. Any four of:
- The linked list is empty i.e. root_ == nullptr, or
- for all nodes n1, n2: (n1 != n2 => n1.user_ != n2.user_) i.e. no duplicates
- for all nodes n1, n2: (n1.next_ == n2 -> n1.user_.id() < n2.user_.id()) i.e. list is in order by id()
- There is no more than 1 edge between any distinct pair of users in the graph.
- no (User*) is a nullptr,
- all edges are symmetric
- user doesn't have edge to itself

b) [5 marks] You are given the following implementation of the Graph method that adds a User.

```cpp
void Graph::add(User * user) {
    if ( find(user) ) throw Graph::DuplicateUserException( user->id() );
    if ( root_ == nullptr ||
        (root_ != nullptr && user->id() < root_->user_->id()) )
        Node * tmp = new Node( user, root_ );
        root_ = tmp;
    return;
    for (Node * ptr = root_; ptr != nullptr; ptr = ptr->next_ ) {
        if ( ptr->next_ == nullptr ||
            (ptr->next_ != nullptr && ptr->next_->user_->id() > user->id()) )
            Node * tmp = new Node( user, ptr->next_ );
            ptr->next_ = tmp;
            break;
    } // for
} // Graph::add
```

Write the preconditions and postconditions for the method:

// Requires:
user != nullptr (or valid user)

// Modifies:
graph/set of users/ this; lost 0.5 if mentioned root_ since only modified if graph is empty, or gets inserted at head of list

// Throws:
Graph::DuplicateUserException if the user is already in the graph

// Ensures:
graph = graph@pre + user , with user inserted in order by user->id()
-1 if doesn't mention order/contents preserved

// Returns:
nothing or N/A or blank

c) [2 marks] Write an abstraction function for the Graph.

```cpp
s = ( ptr == nullptr ? {} : ptr->user_ ) ∪ ptr->next_ 
```

e = ( ptr == nullptr ? {} : ptr->user_.e ) ∪ ptr->next_ , or e is set of edges for that user
-2 if just gave abstraction and representation, but not the function to go between the two
-1 if just referred to linked list mapping to vertices and not to users since we're talking about set of users
Part C: Q4: [45 marks] Coding

See the reference sheet for the definition of the Node and Graph classes.

a) [23 marks] Implement the move assignment operator for the Graph:

```cpp
Graph & Graph::operator=( Graph && other ) {
    if ( this == &other ) return *this;

    // Need to ensure original data is deleted, so either directly swap with contents of "other", or swap with a local variable created on the run-time stack. Second approach is preferable since "other" is thus left in an empty state when otherwise it has the contents of "this".
    Graph tmpGraph;
    std::swap( tmpGraph.root_, other.root_ );
    std::swap( this->root_, tmpGraph.root_ );
    std::swap( this->root_, other.root_ );

    return *this;
}
```

b) [2 marks] Right now, the Graph will leak memory if any of its operations throw/raise an exception. Rewrite the definition of the Graph class data member root_ to ensure that it will always be cleaned up, no matter how the program exits.

Simplest answer is to wrap the root_ in a smart pointer, such as unique_ptr or shared_ptr. Otherwise, need an intermediate object allocated on the run-time stack that holds root_.

Version 1:

```cpp
std::unique_ptr<Node> root_; // or shared_ptr
```

Version 2:

```cpp
struct NodeWrapper { // 1 mark for the class/struct and destructor
    Node * ptr_;
    NodeWrapper( Node * ptr = nullptr ) : ptr_(ptr) {} ~NodeWrapper() { delete ptr_; }
};

NodeWrapper root_;
Part C: Q5: [19 marks] Design Patterns

a) [7 marks] People (bettors) place bets upon horses in races to either win, place, or show. Modify the given UML diagram to incorporate the Observer design pattern so that a bettor can be notified when a race is over. Make sure that your diagram includes all necessary data attributes, operations, visibilities, multiplicities and constraints.

[0.5] Subject superclass
[0.5] Observer superclass

[1] Subject and Observer are abstract base classes i.e. names are italicized or uses {abstract}
[0.5] Race inherits from Subject (correct arrowhead, in correct direction)
[0.5] Bettor inherits from Observer (correct arrowhead, in correct direction)
[1] Observer has pure virtual (italicized) update method
[1] Subject has subscribe/unsubscribe non-pure-virtual methods
[1] Subject has private (or protected) non-pure-virtual notify method
[1] Subject has aggregation (hollow diamond) of Observer with open arrowhead on Observer end and multiplicity of *

-0.5 if no parameters for subscribe/unsubscribe
-0.5 if only provided subscribe but not unsubscribe
Part C: Q5: [19 marks] Design Patterns

b) WatMazon uses the Template Method design pattern to ensure that the customer shipping order follows the proper structural format, which includes printing:

- the item description, quantity and price, and item total, one item per line,
- the total cost of the order before adding on the shipping cost,
- the shipping cost,
- the 13% HST on the order subtotal, including the shipping cost,
- the overall total, and
- the estimated reception date range, based upon the customer's type of order.

For example, for an order of type FreeShipping, we would print:

Order date: 2017/06/24
Widget 3x$0.95 = $2.85
Gadget 1x$1.17 = $1.17
Subtotal: $4.02
Shipping fee: $0
HST: $0.5226
Total: $4.5426
Expected reception: 2017/07/21-2017/08/04

The UML structure is shown below:
Part C: Q5: [19 marks] Design Patterns

i) [8 marks] Write the implementation of `Order::printBill()`, assuming that it prints to `cout`. Don’t worry about formatting the information, just make sure each part of the bill is clearly labelled in the output. See the previous page for the format.

```cpp
void Order::printBill()
{
    double total = 0.0;
    Date orderDate = Date::today();
    cout << "Order date: " << orderDate.toString() << endl;
    for (auto it = items_.begin(); it != items_.end(); it++)
    {
        cout << it->desc() << ' ' << it->quantity() << "x$" << endl;
    }
    cout << "Subtotal: $" << total << endl;
    cout << "Shipping fee: $" << shippingFee() << endl; // Could use a temp var to hold fee to avoid calling twice
    total += shippingFee();
    cout << "HST: $" << total*0.13 << endl;
    total *= 1.13;
    cout << "Total: $" << total << endl;
    cout << "Expected reception: " << orderDate+minNumDays() << " - " << orderDate+maxNumDays() << endl;
}
```

ii) [4 marks] Write the class declaration and definition for the class `FreeShipping`. While the cost of shipping is free, delivery will take a minimum of 4 weeks (28 days), and a maximum of 6 weeks (42 days). (You do not need to put in header guards, using directives or include statements.)

```cpp
class FreeShipping :
    public Order {
        protected:
            int shippingFee() { return 0; }
            int minNumDays() { return 28; }
            int maxNumDays() { return 42; }
    }
```
Use the rest of this page, if necessary, to complete your answers. Make sure you note on the original page that more of your solution can be found here, and be sure to label your work here with the appropriate question number.
*** END OF EXAM ***

Use the rest of this page, if necessary, to complete your answers. Make sure you note on the original page that more of your solution can be found here, and be sure to label your work here with the appropriate question number.