UNIVERSITY OF EDINBURGH

FACULTY OF SCIENCE AND ENGINEERING

MASTER OF SCIENCE IN COMPUTER SCIENCE

PROGRAM AND ALGORITHM DESIGN

Wednesday 25 April 2001

09:30 to 11:00

Examiners:

I. Watson (External)

R.N. Ibbett (Chairman)

INSTRUCTIONS TO CANDIDATES

Answer any TWO questions.

All questions carry equal weight.

1. (a) i. When designing a class in an OO language, what is the principle of *data encapsulation*?

   ii. How is encapsulation achieved using access controls in C++?

   iii. What is the main advantage of encapsulation from the point of view of the implementer of a class? Give a simple example.

   iv. What is the main advantage of encapsulation from the point of view of the user of a class? [*7 marks*]

   (b) i. In an OO language, what is *late binding*?

   ii. Using an example with `Circle` and `Rectangle` subclasses of a `GraphicsObject` class, and a `draw` member function, illustrate why late binding is desirable. Explain what would happen if early binding were used instead.

   iii. How is late binding indicated in the C++ syntax? [*7 marks*]

   (c) i. What does it mean to say that a heap is *fragmented*?

   ii. What is one strategy that an allocator can use to reduce fragmentation? [*4 marks*]

   (d) i. Give two examples of when objects are automatically copied in the execution of a C++ program.

   ii. When is it likely that you have to explicitly define a *copy constructor* for a class? Give an example.

   iii. Copy assignment operators involve overwriting old object values. What subtle problem do you have to guard against here? [*7 marks*]

2. (a) What is the big-O time complexity for *testing for membership of* an integer in

   i. a `vector<int>` object (vector sorted)

   ii. a `list<int>` object (list sorted)

   iii. a `multiset<int>` object [*3 marks*]

(b) Given an iterator pointing to some item of type T in a container object, what is the big-O time complexity for *deleting* that item if the container is

   i. a `vector<T>` object

   ii. a `list<T>` object

   iii. a `multiset<T>` object [*3 marks*]

(c) i. Describe informally the *mergesort* algorithm.

   ii. Give an argument as to why the algorithm terminates.

   iii. Let $T(n)$ be the runtime of mergesort on input of size n. Write down recurrence equations for $T(n)$ that when solved give the time complexity of mergesort. Invent constant factors (e.g. k_1, k_2) as needed. Explain briefly where the different terms in the recurrence equations come from. [*7 marks*]

(d) Consider a *symbol table* ADT (or *associative container* in STL terminology) storing key value pairs.

   i. What are the two most basic operations that a symbol table ADT is expected to support?

   ii. One implementation of a symbol table involves binary search trees. What is the invariant property maintained in a binary search tree?

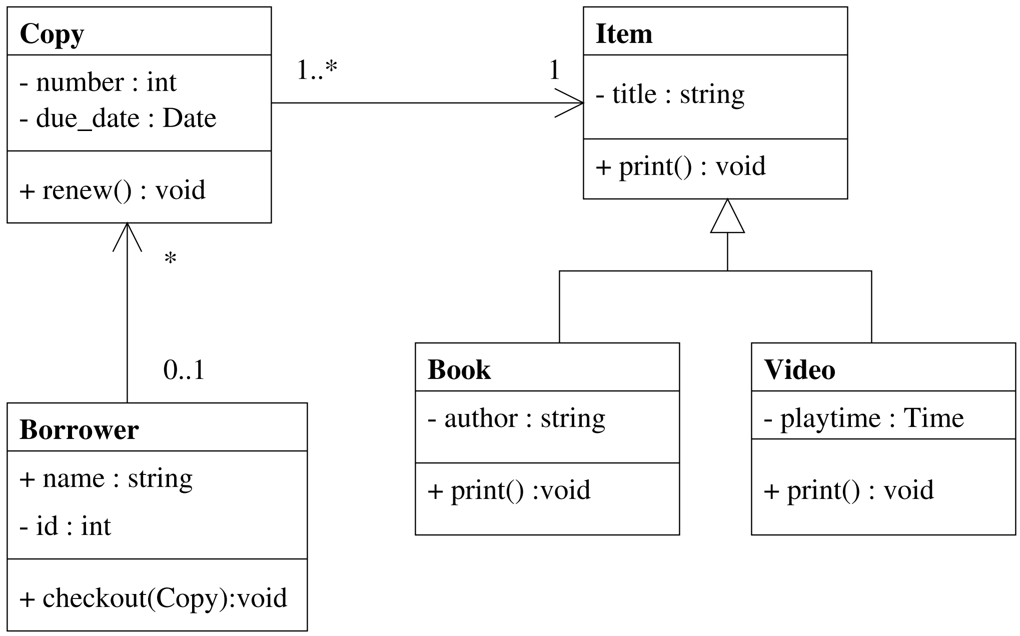
   iii. Discuss the time complexity of the basic operations identified above on binary search trees. [*6 marks*]

(e) i. In the STL, what is a *function object*?

   ii. Give an example of a *function object adaptor*. What does it do?

   iii. What is a *closure* of a function? [*6 marks*]

3. (a) Consider the UML Class Diagram below for a simple library system which keeps track of who has borrowed which copies of books and videos.



i. Explain what the '1..*' notation on the arrow from the **Borrower** box to the **Copy** box indicates.

ii. Sketch C++ class definitions that would implement this diagram. Use the notation '`{...}`' . for writing member function bodies.
If appropriate, use extra classes from the C++ Standard Template Library. Assume the **Date** and **Time** classes have already been defined. Assume that **Item**, **Copy** and **Borrower** objects all have independent lifetimes. Make **Item** an abstract base class, and assume that the **Item** class hierarchy might be extended at some future stage.
Features that will be looked for in your code include:

- member function and instance variable declarations,
- access controls,
- implementation of associations,
- inheritance,
- appropriate use of early/late binding,
- class definition ordering.

You will not be penalised for minor syntax errors. [*15 marks*]

QUESTION CONTINUES ON NEXT PAGE

(b) A class library is being designed with requirements for two stack classes:

- `Stack`: This class has the usual methods such as `pop`, `push` and `is_empty`. Stacks represented by objects in this class are permitted to grow to arbitrary size.
- `BoundedStack`: Stacks represented by objects in this class have a maximum size which is set when the objects are created. This class has the same interface as `Stack` plus a constructor taking a size argument and an extra method `is_full`. The `push` method is expected to generate an exception if an attempt is made to add a new item to a full bounded stack.

It is expected that both classes will be used as parameters to container classes.

Two proposals are considered for the relationship between the `Stack` and `BoundedStack` classes:

1. `BoundedStack` should be implemented as a subclass of `Stack`.
2. `Stack` should be implemented as a subclass of `BoundedStack`.

Discuss and compare the proposals under the following headings:

- Possibilities for code reuse.
- Additional methods needed in `Stack` class.
- Properness of subtyping.

[*10 marks*]