CS_M71 2005/06
Design Patterns and Generic Programming

(Attempt 2 questions out of 3)

Question 1 Lifetime of objects, and exception-safety

(a) Construction and destruction of objects

  (i) What are the "special member functions" of a class (in C++), and what is their purpose (consider each case on its own, and try to be as specific as possible).

  [8 marks]

  (ii) Explain the possibilities for the "storage duration" of an object (in C++), and how the storage duration affects the destruction of objects.

  [4 marks]

  (iii) Explain the "Resource Acquisition Is Initialisation" (RAII) idiom. Discuss the advantages arising from the use of RAII. Consider the example of the connection to a database: What happens if we cannot close the connection?

  [7 marks]

(b) State the different levels of exception safety, and explain the meaning of these levels.

[6 marks]

Question 2 Classes and functions

(a) Classes

(i) List the access specifiers for classes available in C++, and explain how they restrict access to members. Explain in general terms which members should be public and which should be private or protected. Give examples for each case.

[**6 marks**]

(ii) Discuss in general terms the assumptions on the relations between a public base class `B` and a derived class `D`. Give examples when deriving `D` from `B` can be misleading (you might use the "Square vs. Rectangle" discussion).

[**7 marks**]

(b) Freestanding functions

(i) Discuss when to use member functions and when to use freestanding functions (in C++). Explain how "argument-dependent name lookup" (ADL) helps to establish the "Interface Principle" (IP).

[**6 marks**]

(ii) Explain how "tagging classes" are used to avoid a combinatorial explosion of concepts, and how we can equip global functions to be aware of this kind of "tagging polymorphism".

[**6 marks**]

Question 3 Generic programming and polymorphism

(a) Consider the task of comparing one sequence of objects with another sequence of objects. Design a generic algorithm `equal` for this task, using iterators and making as few assumptions as possible. Try to be as precise as possible about the conceptual requirements on the iterators. (Using iterators means that you do not have access to the whole containers; you can only use the "pointers" into them.)

[10 marks]

(b) Forms of polymorphism

(i) Give an example for the visitor design pattern, and explain in which way the visitor design pattern can be regarded as a "transposition" of the usual organisation of virtual member functions in a class hierarchy.

[10 marks]

(ii) Compare the use of templates and polymorphic classes (in C++). Under what circumstances do we really need to use polymorphic classes ("subclass polymorphism") ?

[5 marks]