Concurrent Systems and Applications

(a) Mutual exclusion is an important consideration in many multi-threaded processes.

(i) Describe the syntax and semantics of each of the different ways of using the \texttt{synchronized} keyword in Java. [5 marks]

(ii) What is a \textit{re-entrant} mutual exclusion lock and why is it helpful that the locks used by the Java Virtual Machine to implement synchronized methods be re-entrant? [2 marks]

(iii) Accesses to fields of most data types in Java are atomic but some are not. Give an example of a field access that is not atomic and explain how read and write access can be achieved in a thread-safe fashion. [2 marks]

(iv) Recent editions of the Java language include \textit{generics}. What is the scope of the mutual exclusion caused by the use of the \texttt{synchronized} keyword in a generic class definition? [1 mark]

(b) \textit{Deadlock} can occur in multi-threaded applications.

(i) What four properties hold when deadlock exists? [4 marks]

(ii) Which of these are properties of the Java language and which depend on the program being executed? [1 mark]

(iii) A practical strategy to avoid deadlock is to enforce an ordering on acquiring locks. Explain how this ensures that deadlock is never possible. [2 marks]

(c) Mutual exclusion locks are language-level features. Explain how they can be implemented in terms of either \texttt{Counting} Semaphores provided by the operating system or atomic compare-and-swap operations provided by the hardware. [3 marks]