1. **(a)** What is meant by the time complexity of an algorithm?

In particular, what is meant when we say that a particular algorithm's time complexity is $O(n)$? $O(\log n)$? $O(1)$?

(4)

(b) Recall that a *list* is an indexed sequence of elements. How you would represent a list (i) by an array and (ii) by a linked-list?

Illustrate your answer by showing both representations of the following lists of words:

«'time', 'flies'»
«'I', 'think', 'therefore', 'I', 'am'»

(4)

(c) Assuming (i) the array representation of lists, and (ii) the linked-list representation of lists, as in your answer to part (b), briefly outline algorithms to do the following:

- fetch the element at index i of a list (e.g., if $i = 0$ then the answer should be the list's first element);
- insert a new element at index i of a list (e.g., if $i = 0$ then the new element should be inserted before the list's first element);
- insert a new element at the end of a list (i.e., after the list's last element).

Also write down the time complexity of each algorithm.

(9)

(d) What factors would you take into account in choosing between arrays and linked-lists to represent lists in a particular application?

(3)

2. **(a)** You are given the following requirements for a *stack* abstract data type:

- It must be possible to make an existing stack empty.
- It must be possible to push a given element on to a stack.
- It must be possible to pop the topmost element from a stack.
- It must be possible to test whether a stack is empty.

Write a contract for a *homogeneous* stack abstract data type, expressing your contract in the form of a Java generic interface with suitable comments.

(5)

(b) Briefly describe a possible representation for a stack.

(3)

(c) Consider *stack-machine code* consisting of the instructions summarised in Table 1. Each of these instructions acts on a stack of integers.

Any integer expression can be translated to such stack-machine code. After the code is executed, the stack will contain a single integer, which is the result of evaluating the expression. For example:

Expression	Stack machine code	Expected result
$1 - (3 \times 4)$	LOAD 1; LOAD 3; LOAD 4; MUL; SUB	–11
9 + (6 / 3) – 5	LOAD 9; LOAD 6; LOAD 3; DIV; ADD; LOAD 5; SUB	6
(9 + 6) / 3 – 5	LOAD 9; LOAD 6; ADD; LOAD 3; DIV; LOAD 5; SUB	0

Draw diagrams showing the contents of the stack after executing each instruction in the stack-machine code "LOAD 1; LOAD 3; LOAD 4; MUL; SUB". Assume your stack representation of part (b).

(4)

(d) Assume the following representation of stack-machine instructions:

```
public class Instruction {
  // Each Instruction object is a stack-machine instruction.

  public byte opcode;  // LOAD, ADD, SUB, MUL, or DIV
  public int operand;  // used only if opcode = LOAD

  public static final byte LOAD = 0,
      ADD = 1, SUB = 2, MUL = 3, DIV = 4;
}
```

In terms of your stack contract of part (a), implement the following Java method:

```
static int execute (Instruction[] instructions);
// Execute the stack-machine code in instructions, and return
// the result.
```

(8)

Table 1 Stack machine instructions (Question 2).

Instruction	Effect
LOAD *i*	Push the integer *i* on to the stack.
ADD	Pop two integers off the stack, add them, and push the result back on to the stack.
SUB	Pop two integers off the stack, subtract the topmost integer from the second-topmost integer, and push the result back on to the stack.
MULT	Pop two integers off the stack, multiply them, and push the result back on to the stack.
DIV	Pop two integers off the stack, divide the second-topmost integer by the topmost integer (discarding any remainder), and push the result back on to the stack.

3. **(a)** In the context of abstract data types, define what we mean by a *set*.

(2)

(b) You are given the following (simplified) requirements for a set abstract data type:

- It must be possible to make an existing set empty.
- It must be possible to add a given element to a set.
- It must be possible to remove a given element from a set.
- It must be possible to determine the cardinality of a set.
- It must be possible to test whether a given value is a member of a set.

Write a contract for a *homogeneous* set abstract data type, expressing your contract in the form of a Java generic interface with suitable comments.

(5)

(c) Explain how a set can be represented by a *closed-bucket hash-table* (CBHT).

Illustrate your answer by showing how the set of dates {05/11/1946, 06/06/1962, 05/11/1964, 05/05/1978, 23/02/1983, 01/01/2007} would be represented by a CBHT with 13 buckets and the following hash function:

hash(*date*) = month number of *date*

(6)

(d) Consider an application that uses a set of about 100 dates. Explain why hash function of part (c) would be a poor choice in this application.

Suggest a better hash function, and explain why it is better.

(7)

4. **(a)** Write a short account of the Java Collections Framework (200–300 words). Your account should cover the most important collection classes and interfaces. Illustrate your account with a class diagram showing how these classes and interfaces relate to one another.

(*Note:* You should describe each class and interface briefly, mentioning only its most important features. You need not cover iterators and comparators.)

(12)

(b) The Java Collections Framework does not support *graphs*. How might the Framework be extended to support graphs? Illustrate your answer by extending your class diagram of part (a).

(8)