Answer ALL questions from Part I and TWO questions from Part II

Part I

1. a) Java container classes store references of type Object. Explain why.
   What are the consequences for clients of containers?

[4 marks]

   b) The Java class ArrayList uses a primitive array as its internal data structure and provides methods to access the array elements. Outline the Java implementation of a basic version of ArrayList, providing methods to get, set, insert and delete data values. It should be possible to insert and delete at any position within the ArrayList. The methods should throw exceptions as necessary. Include one or more appropriate constructors.
   (Exact Java syntax is not required, pseudocode is acceptable for method implementations.)

[14 marks]

   c) What is an iterator? Outline how an iterator class would be implemented for your ArrayList class.
   (Exact Java syntax is not required, pseudocode is acceptable for method implementations.)

[6 marks]

   d) Outline a test plan for your ArrayList and iterator classes.

[6 marks]

[Total 30 marks]

2. a) The following two augmented grammars are expressed in yacc:

```
list   :  list  INT  {print($2);}
       |  INT
```

```
list   : INT list {print($1);}
       |
```

i) From each grammar derive a parse tree for the sentence 1 2 3 (which when tokenised becomes INT INT INT). By traversing that tree show the results of executing the print actions.

ii) Explain why, when using yacc, left recursion may be preferred for the definition of simple lists.

iii) Explain why left recursion is appropriate for the definition of simple arithmetic expressions.

iv) Left recursive grammars are not appropriate as a basis for the construction of recursive-descent parsers. Explain why this is so.

v) Give another instance of a circumstance in which a left-recursive definition would not be appropriate.

[8 marks]

Question 2 continues on the next page

CONTINUED

b) The following SR-table was derived from the output produced by yacc from the input:

```
%%
E : E E '+'
E : 'e'
```

input / state	ACTION			GOTO
	e	+	$end	E
0	s2			1
1	s2		accept	3
2	r2	r2	r2	
3	s2	s4		3
4	r1	r1	r1	

i) Explain how the entries s2 and r2 are to be interpreted. Also explain how a blank entry is to be interpreted.

[3 marks]

ii) Show the moves which the parser would make when parsing the sentence:

e_1 e_2 + e_3 +

[5 marks]

iii) Construct the corresponding parse tree, labelling each node with the number of the corresponding move from the parse.

[4 marks]

[Total 20 marks]

END OF PART I

TURN OVER

Part II

Answer Two questions

3. Suppose it is desired to add a new construct to Java, a *repeat* statement, which executes a statement a number of times. The number of iterations is determined by evaluating an expression. Thus

```
repeat (20)
        System.out.print("*");
```

would call print twenty times.

The following code would also call print twenty times:

```
n = 10;
repeat (2*n) {
        System.out.print("*");
}
```

a) Show the kind of target code for execution on a stack-based machine which might be generated from the second example. Translate 'System.out.print("*");' as 'print "*"'. Hint: consider using the top of the stack to store the value of the expression; you may then need to use *copy* and *pop*, if so explain why.

[6 marks]

b) Draw a syntax diagram for the *repeat* construct. You may assume that *expression* and *statement* are already defined.

[3 marks]

c) Annotate your syntax diagram to generate code for a stack machine.

You will need to make some choices about the semantics of your implementation. Make these choices explicit.

[8 marks]

d) Briefly describe how the generation of labels would be handled in a method implementing your annotated syntax diagram.

Your implementation should work for nested constructs; briefly justify that it does.

[4 marks]

e) Suppose that in the body of the second *repeat* example above an additional statement, the assignment n = 0; was inserted after the *print* statement. Explain what would happen when the *repeat statement* was executed.

[4 marks]

[Total 25 marks]

CONTINUED

4. a) What is a thread?

[2 marks]

b) Explain how a thread is created and started.

[4 marks]

c) Explain how event handling works when a Graphical User Interface (GUI) is used. What role does the GUI thread play?

[8 marks]

d) Consider a GUI for a text editor program that includes a menu bar. Outline how the GUI would be constructed using Java Swing components.

[11 marks]

[Total 25 marks]

5. a) Explain what a design pattern is, giving an example. What are the key advantages of using design patterns?

[7 marks]

b) What are the roles of interfaces and abstract classes in Java? How does the programmer decide which to use?

[7 marks]

c) Consider the problem of writing seat booking applications for a range of organisations such as theatres or train companies. Identify a collection or framework of abstract classes that could be reused across different specific booking applications. Construct a UML class diagram showing the classes and their relationships.

[11 marks]

[Total 25 marks]

END OF PAPER