The purposes of questions 1 and 2 are:

1. to remind you of some basic Prolog programming techniques;
2. to test your understanding of some of advanced Prolog features;
3. to compare and contrast the imperative programming language and declarative programming language.

Question 1 [25 Marks]

(i) Enter the OU Prolog system and edit the file PARTY by adding the following fact to the database as the last likes clause:

likes(cyril, philosophy).

Exit from the Editor and obtain a printer listing of the database.

(a) On the print-out of the PARTY database, indicate one example of each of the following:

- the head of a clause;
- the body of a clause;
- a relation having both fact and rules;
- a procedure.

Parts (b), (c) and (d) are concerned with the query:

?- atparty(cyril), compatible(cyril, A), atparty(A).

(b) Describe the procedure that the interpreter would follow when matching the second goal of this query, i.e. compatible(cyril, A), with a clause in the PARTY database, after the first goal, atparty(cyril), has been satisfied. Confine your answer to the matching process alone.

[3 marks]

(c) Draw a box diagram, similar to that shown in Unit 10, Figure 8.6 (page 26), showing how the query is processed. Indicate where backtracking (if any) occurs. What value is given to A?

[5 marks]

(d) Indicate, on your diagram for part (c), using a different colour or dotted lines, what would happen if the query were to be resatisfied. Apart from the resatisfaction itself, indicate any instances of backtracking that take place.

[4 marks]

(ii) Give two positive integers, A and B, their greatest common divisor D, can be found according to three cases:

(a) If A and B are equal then D is equal to A.
(b) If A > B then D is equal to the greatest common divisor of B and the difference of A – B.
(c) If B > A then do the same as in case (b) with A and B interchanged.

Write Prolog predicates gcd(A,B,D) to express the above problem. Enter the OU Prolog system and create a file, says gcd.txt, to test your program. Print out the program and the results for the following 5 queries.
Questions 2 [25 Marks]

Languages used for logic programming are called declarative programming languages because programs written in them consist of declarations rather than assignments and control flow statements as in imperative programming languages.

(i) Prolog is a declarative programming language while Pascal is an imperative programming language. Compare and contrast Prolog and Pascal in terms of

(a) data type
(b) data manipulation
(c) program structure
(d) program control

(ii) Is the order of clauses in a Prolog program important? Justify your answer by discussing the effect of order of clauses in the following Prolog program.

positive_integer(0).
positive_integer(N) :- positive_integer(M), N is M + 1.

Hint: What if the order of clauses in the above program is changed as follows:

positive_integer(N) :- positive_integer(M), N is M + 1.
positive_integer(0).

Can we say Prolog is an ideal interpreter? If not, what can we say for this kind of interpreter?

(iii) This section tests your understanding of the predicates repeat and fail.

(a) Briefly describe the Prolog predicate repeat and fail.

(b) Indicate the effect of the predicate fail in the following program:

book(physics,duncan).
book(chemistry,cclee).
find_all_books :-
    book(Title,Author),nl,
    write(Title),
    write(‘ is written by ’),
    write(Author),
    fail.
(c) Describe the difference in the program and results for the following 2 programs (Refer to Unit 9, Section 5.3):

program1 :-
    repeat,
    write('Enter an animal name'), nl,
    read(X),
    special(Y,X),
    write(Y), nl,
    fail.

program2 :-
    write('Enter an animal name'), nl,
    read(X),
    process(X).

process(end).
process(X) :-
    special(Y,X),
    write(Y), nl,
    program2.

[6 marks]
Question 3 [25 Marks]

This question is based on Units 11.

Here is a BNF definition of a grammar with <expression> as the start symbol.

<expression> ::= <statement>

<statement> ::= <value> | <statement> <operator> <statement>

<operator> ::= + | *

<value> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 0

(i) Show that the above grammar is ambiguous by demonstrating that it allows the recognition of the string 2 + 4 * 3 + 7 as an <expression> with (at least) two different phrase structures. Your answer should give, for two different phrase structures, the appropriate syntax tree and the appropriate expression tree (in a manner similar to that given on page 12 of Unit 11).

8 marks

(ii) For each of the phrase structures that you produced in part (i), write down a parenthesized form (i.e. with brackets) of 2 + 4 * 3 + 7 which indicates the order of evaluation implied by the phrase structure.

4 marks

(iii) Amend the grammar to remove the ambiguity by giving the + operator precedence over the * operator.

4 marks

(iv) Draw the tree structure representing the phrase structure of 2 + 4 * 3 + 7 when recognized as an <expression> in your amended grammar.

2 marks

(v) State whether your amended grammar contains

(a) left-recursion;

(b) right-recursion.

Justify your answer.

4 marks

(vi) The original grammar is an LL(K) grammar. What is the value of K? Use the expression in part (i) above to justify your answer.

3 marks

End of Question 3
Question 4  [25 Marks]

This question is based on Units 12.

Unit 12 is built around a small, simple language called SL. Towards the end of the Unit (page 41) two desirable extensions to SL are listed. In this question you are required to deal with these two extensions – while and compound statements.

(i) A <while> statement is to be added to SL. This new statement is to be identified using the % symbol, and should be capable of repeating precisely two statements while the condition (an <expression>) is true. Like the <loop> statement, the <while> statement should be written on one line. The syntax of <while> is

<while> ::= % <expression> ; <statement> ; <statement>

(a) Write a program in SL, using the <while> statement to read in a series of numbers from the keyboard. A number zero is used to terminate the input. The output is the largest of the numbers.

(b) Define the rules for the translation of the <while> statement into IC. Write your answer in the style of Figure 9.5 (page 39). You may omit the syntax, but include the other headings.

(ii) Add the rule for the non-terminal <compound-statement>, to the syntax of SL given in Section 6, using the terminal symbol [ and ], to mark the block beginning and end respectively.

Statements within a compound statement should be separated by <newline>.

Change the rules for the <loop> and <conditional> non-terminals so that they use your new <compound-statement> non-terminal.

In your answer, give only those rules which are new or have been changed.

(iii) Write a program in your extended SL, using the redefined <loop> statement in part (ii) to read in 3 integers from the keyboard, add the three integers and store the sum in a variable $s$. Use the variable $c$, $n$, and $s$ as shown in the following Pascal-like pseudo code:

```
Initialize s to 0;
For c from 1 to 3
BEGIN
   read integer from keyboard into n;
   s := s + n;
END
```

(iv) Illustrate how these new and amended rules work by translating your SL program from part (iii) into an intermediate code program. Use the following symbol table.

<table>
<thead>
<tr>
<th>symbol</th>
<th>location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s$</td>
<td>1</td>
</tr>
<tr>
<td>$c$</td>
<td>2</td>
</tr>
<tr>
<td>$n$</td>
<td>3</td>
</tr>
</tbody>
</table>

[4 marks]

[5 marks]

End of Question 4

End of TMA 03 Questions