Symbolic Computation
Example Test Questions for Prolog

Test questions can be based on the following sources: (1) the textbook(s), (2) readings, (3) lectures, (4) reports, (5) exercises, and (6) on-line notes and slides. They are based on topics from the beginning of the year up to the class before the test.

Consider all concepts and terminology used in the text book, reports, slides and classes and ask the typical questions - how, why, when, where and what - individually and in combination. In particular, variations are based on "describe", "explain", "define", "what is meant by", etc. The shorter programming exercises in the example programming exercises have been and may be asked as test questions.

Many of the following questions have appeared in previous year's tests and examinations. The list is by no means exhaustive.

1. Basic notions

1. Describe what it means to program in Prolog.

2. Describe the prolog data structures, including syntax.

3. Describe of the syntax of Prolog.

4. What is the structure of a program in Prolog?

5. What does it mean to execute a program in Prolog?

6. Prolog lists and Lisp lists are very similar in structure. Explain how they are similar? How they are different? How is dot-notation related to both list structures?

7. What is the purpose of the cut in Prolog?

8. Trace the following fibonacci program with the query fib(3,F).

```
fib(0,1).
fib(1,1).
fib(N,F) :- N1 is N-1, N2 is N-2,
            fib(N1,F1), fib(N2,F2),
            F is F1+F2.
```

9. How can a general tree (no limit to the number of descendants for any node) be represented in Prolog?

10. Explain/define with respect to Prolog the terms: term, rule, resolution, matching, clause, unification, list, Horn clause, closed world assumption, backward chaining, forward chaining, backtracking, functor, compound term, predicate.

11. Lisp has the functions `car', `cdr' and `cons' but Prolog does not. Explain how Prolog can still do the same list processing operations as Lisp.

12. How is Prolog related to data bases?

13. Explain how arithmetic is done in Prolog?

14. How can Prolog be used to construct Prolog rules and use them in a computation?

15. Programs require the notions of sequence, decision and iteration. Explain how these notions are represented in Prolog.

16. How would multi-dimensional arrays be implemented in Prolog? Define an operation 'index' which has an array and an index list as parameters; the function is to return the indexed array element. There is no fixed size for the number of dimensions.

17. Describe a general template for a Prolog predicate that counts with an accumulator.

18. The not predicate can be represented in Prolog as follows.

```
not(P) :- call(P) , ! , fail.
not(_).
```

Show how cut can be represented with not.

19. Prolog provides a means to define operators using op(X,Y,Z). For example, operators 'has' and 'isa' can be defined so the following syntax could be used to enter rules.

```
Animal has hair :- Animal isa mammal.
```

Give an example definition for 'has' and 'isa', then define and explain what the parameters to op are, what general values we can give them and why we choose particular values.

2. Underlying Logic

1. Explain what is meant by Skolemization

2. Using propositional calculus, explain the resolution method for proving theorems.

3. Explain what unification means when matching patterns.

4. Describe resolution method theorem proving? What is its relationship to Prolog?

5. Why is Prolog restricted to Horn clauses?

6. Why is negation, or the not predicate, unsound in Prolog?

7. Skolemization is the process of removing the first order predicate calculus quantifiers 'for every' and 'there exists'. With some examples show what skolemization does.

3. Grammar Rules

1. The parse rules in the chat program were entered in a user friendly form, for example, the following.

```
parse(Clause) --> [ Who , is , the ] , type(T) , [ of ] ,
                  thing(Name) , [ '?' ] ,
                { Goal =.. [T,X,Name] , Clause = '?w'(Goal) , ! }.
```

Such rules are translated into standard Prolog syntax. Give a translation of the above rule and explain how the translation parses a sentence and what semantic actions take place.

2. What are Prolog grammar rules? How are they related to Prolog proper?

3. Describe and explain how the chat program works. For readline describe the basic steps and data structures involved – Prolog is not necessary. For parse and respondTo describe and explain the basic syntax, semantics and data structures used to handle facts and queries.

```
chat :- repeat
        , readLine(Sentence)
        , parse(Clause , Sentence , _ )
        , respondTo(Clause)
        , Clause = stop .
```

4. For the parse and respondTo predicates in chat describe the syntax and explain the basic semantics and data structures used to handle facts and queries.

5. Describe and explain how the readline predicate in the chat program works. Describe the steps and data structures involved – Prolog is not necessary