Revision Questions

1. What concepts underpin the following language paradigms:-

a) imperative
b) declarative
c) object oriented

2. Explain what is meant by the terms "compilation" and "interpretation"

3. Explain how the process of compilation can be partitioned into different stages and hence the terms "pass" and "phase" in the context of the compilation process

1. Explain the relationship between the following terms:- "grammar", "language", "set of strings", "alphabet", "string/sentence/word"

2. Identify and explain four criteria for evaluating the definition of a programming language

3. Explain with the aid of simple examples Chomsky's classification of grammars

4. Explain what is meant by the term "ambiguity" in the context of a formal definition, e.g. a formal definition of a programming language's syntax

5. Explain with the aid of simple examples what is meant by the terms "regular language" and "regular expression", "finite state automaton"

6. Explain how a compiler may be organised by a useful separation of related concerns, i.e. how the process of compilation can be divided into distinct stages each of which results in a well-defined output.

7. Explain the following terms:- "source description (or source string)", "lexical analysis (or scanning or itemisation)", "syntax analysis (or parsing)", "semantic analysis (or checking)", "code (or report) generation" and "(code) optimisation".

8. Explain with the aid of simple examples what is meant by the terms "single-pass" and "multiple-pass" in the context of the compilation process

9. Explain with the aid of simple examples what is meant by the terms "top-down" and "bottom-up" parsing in the context of the compilation process.

10. Explain with the aid of simple examples the main function of a lexical analyser (or scanner).

11. Explain what is meant by the term "syntax directed translation" in the context of the compilation process

12. Explain with the aid of a simple example how a Finite State Automata (FSA) can be constructed to recognise the lexemes of a language.

13. Explain what is meant by the terms "Don-deterministic Finite Automaton (NFA)", "regular expression" and "Deterministic Finite Automaton".

14. Explain with the aid of simple examples two alternatives means of organising the parsing *process*, i.e. how Top-down parsing **Derives** the input sentence from the starting symbol 'S' and how Bottom-up parsing **Reduces** the input to the starting symbol 'S'.

15. Explain what is meat by the term "parse tree"

16. Describe the process of "top-down parsing by recursive descent"

17. Use syntax graphs to describe the rules by which a top-down recursive descent parser may be generated

18. Explain the following terms in the context of top-down parsing by recursive descent, "first functions", "semantic actions", "error handling"

19. Explain with the aid of simple examples the terms "S-Grammar" and "LL1 grammar"

20. Explain with the aid of simple examples the terms "First Set" and "Follow set"

21. Explain with the aid of a simple example the term "Top-down Table Driven LL1 Parsing"

22. Explain with the aid of a simple example how an LL1 parse table can be constructed.

23. Explain with the aid of a simple example the term "Bottom-Up parsing" and hence the terms "right-most derivation", "right sentential form", "handle" and "prefix"

24. Explain with the aid of simple examples the terms "Shift-Reduce parsing" and "LR parsing"

25. Explain what is meant by the terms "name list" and "property list"

26. Describe, with the aid of simple examples data structures suitable for implementing name lists

27. Explain what is meant by the term "textual level" in the context of a property list

28. Explain what is meant by the term "block-structured property list" and hence the terms "property entry", "type entry" and "label entry"

29. Explain with the aid of simple examples, what is meant by the terms "hash-table" and "hash-table with chaining"

30. Explain what is meant by the term "declarative processing" in the context of the process of semantic analysis and hence the terms "label declaration", "constant declaration", "type declaration", "variable declaration" and "procedure/function declaration"

31. Explain what is meant by the term "imperative processing" in the context of the process of semantic analysis

32. Explain what is meant by the term "tree-based intermediate representation" in the context of the process of semantic analysis

33. Explain with the aid of a simple example what is meant by the term "syntax tree"

34. Explain with the aid of a simple example what is meant by the term "syntax directed translation"

35. Explain how an instruction set can be classified in terms of the kinds and number of instructions it supports

36. Explain with the aid of simple examples the terms "zero-address" "one-address", "two-address" and "three-address" codes

37. Explain the addressing modes of a typical microprocessor

38. Describe code sequences for simple Pascal constructs involving scalar variables and hence the allocation of storage to simple types

39. Describe simplified M68000 code sequences for Pascal’s and ‘C’ control constructs

40. Explain how, in a simple static allocation strategy, a compiler allocates ‘fixed’ (or static) storage locations for variables in a contiguous ‘block’ of memory.

41. Explain, with the aid of a simple example, the term "dynamic allocation" in the context of the organisation of a run-time stack

42. Explain, , with the aid of a simple example, the term "non-local variable" in the context of the organisation of a run-time stack, and hence the terms "static chain" and "display technique"

43. Explain, with the aid of simple examples, how a compiler allocates memory for structured data items and how instruction sequences can be generated to manipulate such structures.

44. Describe, with the aid of a simple example, how values of an array type whose elements are each record types are arranged on the (M68000) stack.

45. Write an instruction sequence (M68000) for an assignment statement that uses a value of the array type from Q44) above

46. Explain the terms "storage management", "heap management" and "garbage collection"

47. Explain, with the aid of simple examples, what is meant by the term "code generation"

48. Explain, with the aid of simple examples, the terms "assembler", "target code" and "target program"

49. Explain, with the aid of simple examples, the terms "intermediate code generation", and "TAC instruction"

50. Explain, with the aid of simple examples, the terms "code optimisation", and hence how optimisation structures can be classified

51. Explain, with the aid of simple examples, the terms "block optimisation", "iteration (loop) optimisation", "intra-procedural optimisation" and "inter-procedural optimisation"

52. Explain, with the aid of simple examples, the terms "control flow optimisation", "algebraic simplification", "strength reduction"

53. Explain, with the aid of simple examples, th terms "peephole optimisation", "redundant instruction" and "expression optimisation"

54. Explain, with the aid of simple examples, the terms "imperative language paradigm" and "type domain"

55. Explain, with the aid of simple examples, the "type completeness principle", the "correspondence principle" and the "abstraction principle"

56. Construct a top-down recursive descent recogniser for the following grammar:-

```
<concatenation>    ::= <first> <second> <third>
<first>          ::= "@" <digit> | <digit>
<second>          ::= <plain> | <augmented>
<third>          ::= <char> <char>
<char>           ::= "a" | "b" | … | "z"
<plain>          ::= "%" <char>*
<augmented>      ::= "[" <plain> "]"
<digit>          ::= "0" | "1" | … | "9"
```

You may assume that your recognition procedures will be used within the program whose general

structure is shown below:-

```
PROGRAM parser;

  TYPE symbol = (sym_ampersand,  sym_percent,
                 sym_open_paren, sym_close_paren,
                 sym_digit,      sym_char,  ….
                );

  VAR  current: symbol;

  PROCEDURE read_sym(VAR current: symbol);
  BEGIN
    …
  END;

  PROCEDURE error(s: string);
  BEGIN
    …
  END;

  FUNCTION first_first(current: symbol): Boolean;
  BEGIN
    first_first:= (current = sym_ampersand) OR (current = sym_digit)
  END;

  FUNCTION first_concatentation(current: symbol): Boolean;
  BEGIN
    first_concatentation:= first_first(current)
  END;

 …. (* rest of first functions *)

    recognition procedures to be fitted here

BEGIN
  read_sym(current);
  IF first_concatentation(current) THEN rec_concatentation(current)
  ELSE error('concatentation expected')
END.
```

57. How can a compiler be organised so that its structure reflects the process that the compiler must undertake?

58. Describe with the aid of simple examples the processes of "syntax analysis" and "semantic analysis"

59. Explain with the aid of simple examples the relationship between an LL1 grammar and an "equivalent" non-LL1 grammar and hence how the former can be derived from the latter

60. Explain how the Pascal programming language embodies the imperative programming language paradigm