1. Scheme belongs to the class of (mark one, 4 points)
   (a) Object-oriented languages
   (b) Functional languages
   (c) Procedural languages (von Neumann)
   (d) Logical languages

2. Which language(s) are considered object oriented? (mark one or more, 4 points)
   (a) Fortran 77
   (b) Java
   (c) PL/I
   (d) Smalltalk-80

3. Which tool below combines (static) library routines and incomplete object codes into an executable machine language program? (mark one, 4 points)
   (a) Assembler
   (b) Compiler
   (c) Interpreter
   (d) Linker

4. What class of errors are detected by a parser that is part of a compiler? (mark one, 4 points)
   (a) Lexical errors
   (b) Syntax errors
   (c) Semantic errors
   (d) Program errors

5. What is an ambiguous grammar? (mark one, 4 points)
   (a) A grammar for ambiguous languages only
   (b) A grammar augmented with semantic rules
   (c) A parser for an ambiguous grammar cannot construct a unique parse tree for certain inputs
   (d) A parser for an ambiguous grammar produces abstract syntax trees

6. Which of the statements below are true? (mark one or more, 4 points)
   (a) An LL parser is a top-down parser
   (b) An LR(1) grammar of a language can be used to implement a top-down and a bottom-up parser for this language
   (c) An LL(1) grammar of a language can be used to implement a top-down and a bottom-up parser for this language
   (d) Recursive descent parsing is a bottom-up parsing method
7. What is a free format language? (mark one, 4 points)

(a) A language in which indentation is significant, i.e. the amount of spacing is meaningful and influences the execution of a program

(b) A language in which the relative positions of tokens with respect to each other is important rather than the position of the tokens on the page

(c) A language with formatted read and write constructs

(d) None of the above

8. Consider the Scheme function below:

```scheme
(define fun
  (lambda (arg)
    (cond
      ((null? arg) 0)
      ((list? arg) (car arg))
      ((number? arg) (+ arg 1))
      (else arg)
    )
  )
)
```

Which of the following function evaluations are correct? (mark one or more, 4 points)

(a) `(fun 5)` evaluates to 6
(b) `(fun "3")` evaluates to "4"
(c) `(fun '(a b))` evaluates to a
(d) `(fun (fun '()))` evaluates to 1

9. What is an *assembler*? What is it used for? (explain, 7 points)

10. What is a *token*? What are they used for? How are they defined? (i.e. what notation is used to express them?) (explain, 7 points)
11. What is the *front-end* of a compiler and what is its purpose? (explain, 7 points)

12. Give an example of a *higher-order function* in a programming language of your choice. (7 points)

13. What are *static semantic checks*? Give an example of a static semantic error in a programming language of your choice. (explain, 7 points)

14. Write a “Fibonacci function” in Scheme (10 points). The function should implement

   \[ fib(n) = \begin{cases} 
   1 & \text{if } n < 3 \\
   fib(n - 2) + fib(n - 1) & \text{otherwise}
   \end{cases} \]
15. Consider the grammar:

\[ \text{nested} \rightarrow "(" \text{nested} ")" \]
\[ \text{nested} \rightarrow "[" \text{nested} "]" \]
\[ \text{nested} \rightarrow \epsilon \]

(a) Is the sentence ([]) syntactically correct? (yes/no, 3 points)
(b) Is the sentence ([]) syntactically correct? (yes/no, 3 points)
(c) What kind of language does this grammar describe? (7 points)

(d) Draw a parse tree of the input sentence: (())

(10 points)