1. (10 pts) The if statement in Pascal has the syntax:

```pascal
if boolean_expression then statement else statement
```

In Ada, the syntax for the if statement is:

```ada
if boolean_expression then statement else statement end if
```

What are the advantages of introducing an explicit terminator, such as the `end if` in Ada?

2. (20 pts) Write a grammar that describes arithmetic expressions in prefix notation, where possible operators are `+` and `*`, and possible operands are numbers or identifiers. You do not need to specify the internal structure of numbers and identifiers – assume that they are returned by the scanner as terminal symbols `nr` and `id`. Also assume that each operator takes exactly two operands. Is your grammar ambiguous? Why?

3. (20 pts) Suppose that we try to write a short-circuit version of `and` (with two operands) in C as:

```c
int sc_and (int a, int b)
{
    return a ? b : 0;
}
```

Explain why this does not produce a short-circuit behavior. Would it work if normal-order evaluation were used? Why?

4. (30 pts) In the C programming language:

(a) (10 pts) Show how to simulate a do statement (shown below) with a while statement.

```c
do
    s;
while(c);
```
(b) (10 pts) Show how to simulate a while statement (shown below) with a do statement.

```scheme
while(c)
  s;
```

(c) (10 pts) Show how to simulate a for statement (shown below) with a do statement.

```scheme
for(s1;c;s2)
  s;
```

5. (20 pts) Using the Scheme programming language, write a tail-recursive function that returns the sum of all elements in a list of numbers. You will probably want to also define a “helper” function, as shown in Section 6.6.1 of the textbook.

6. (Extra Credit - 10 pts) Show (in low-level pseudo-code, as illustrated in the textbook) what would be the target code generated for the tail-recursive function from problem 5.