To complete this project, you will define a programming language, Language X, which is an extension of Language Three from Chapter 23. Here is a sample program in Language X:

```
let
  val sum = fn x =>
    if null x then 0
    else (hd x) + sum (tl x)
in
  sum (1::2::3::[]) end
```

These are the additions to Language Three, as exemplified in the sample program above.

- The special constant [].
- The operator :: for consing a value onto a list.
- The primitive functions null, hd and tl.
- The if expression.

In precedence, associativity, and behavior, these additions should work like they do in ML. Like Language Three, Language X has no type checking. Like the first definitions of Language Three in Chapter 23, Language X uses dynamic scoping.

The project has four parts, each worth 20 points:

1. Give a BNF grammar for Language X. Make sure it is unambiguous and establishes the correct precedence and associativity. (Hint: start from the BNF for Language Three, on page 508.)

2. Give a Prolog interpreter val x for Language X. (Hint: start from the val3 interpreter for Language Three, on page 508. Note that it uses the lookup predicate on page 504. I suggest that you use special-purpose AST nodes for cons, null, hd and tl. That is, for an application of the hd function, the AST should have a node of the special form hd(Actual), rather than a node of the general form apply(Function,Actual). That will make your interpreter easier to write.)

3. Give a natural semantics for Language X.

4. Show your interpreter operating on the AST for the example program above, producing the correct result.