Test 1

NAME:_____ STUDENT NO:_____

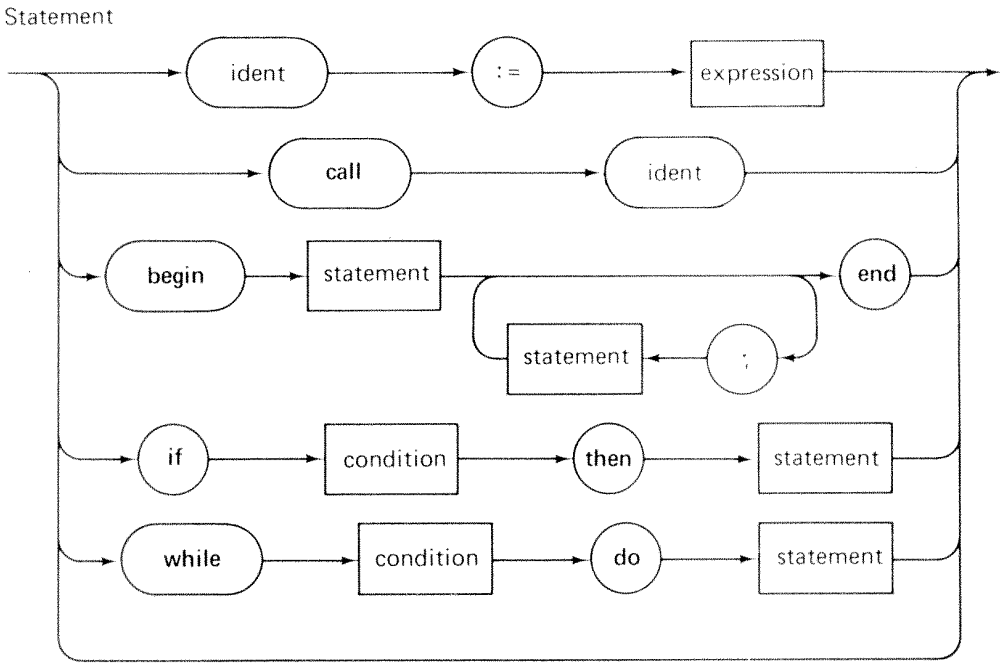
1. (20%)
   You have a native `Java` compiler running on your Pentium (`x86`) machine, i.e., a `Java` to `x86` compiler written in `x86` binary executable. Your friend has just written a `Java` compiler in `Java` for his PowerPC (`PPC`) machine. Explain how you can help your friend to build a native `Java` compiler for his machine.

2. (20%)

   (a) (10%) Other than reserved keywords in a programming language, show *three* examples how *names* are used in a program. For each example, discuss its essential attributes associated with that name.

   (b) (10%) *Recursion* and *lexical scope* are supported by the majority of programming languages in practical use. Discuss a runtime mechanism that is needed to supported these features.

3. (20%) Given the following syntax diagram for a subset of `PL/0`:



(a) (5%) Write down an extended BNF grammar rule for this subset.

(b) (15%) Sketch (in pseudo code) a *recursive decent* parser for this subset, only those specified by this syntax diagram.

4. (20%)

   (a) (6%) What is *overloading*? Discuss its *pros* and *cons*.

   (b) (6%) What is the architecture of a compiler? (You may use `PL/0` compiler as an example.)

   (c) (8%) What is a *compiler-compiler*? Discuss its input and output as compared to a typical compiler.

5. (20%)

   (a) (5%) What is *universal polymorphism*? Illustrate with a simple example.

   (b) (5%) Use `Java` or `C` as a typical programming language, explain with an example the difference between *name* and *structural* type equivalence.

(c) (10%) Given the following type construction notation:

```
<Type> ::=  "int"   |   "bool"   |   "array of" <Type>   | <Var>
           <Type> "-->" <Type>    |    <Type> "x"  <Type>
<Var>  ::=  alpha1  | alpha2  | ...
```

Calculate a *most general type* of the function `f` below. Show your steps. (Note: `size` returns the number of elements).

```
function f  ( a :  ???  ,   b : ??? ) return ??? {
    for ( i = 0; i < size(a); ++i ) {
        c[i] = b( a[i] );
    }
    return c;
}
```