Test 1

NAME:____________________ STUDENT NO:____________________

1. (20%)
   The language `C` is becoming the universal "assembly" language for most compiler writers. It is because you can always find a native or cross `C` compiler for your favourite machine (e.g., `x86`, `ARM`).

   Assume that you are interested in designing and implementing a new programming language `X`. You decided to implement a compiler for this language `X` in `X` but translates its input to the target language `C`. And then invoke the native `C` compiler to generate the machine code afterwards.

   However, you cannot run this compiler yet because it is written in `X` even though it can translate a program in `X` to `C`. You are trying to bootstrap this compiler using your native `C` compiler (e.g., `x86`). Explain how you can do this with *minimal* effort.

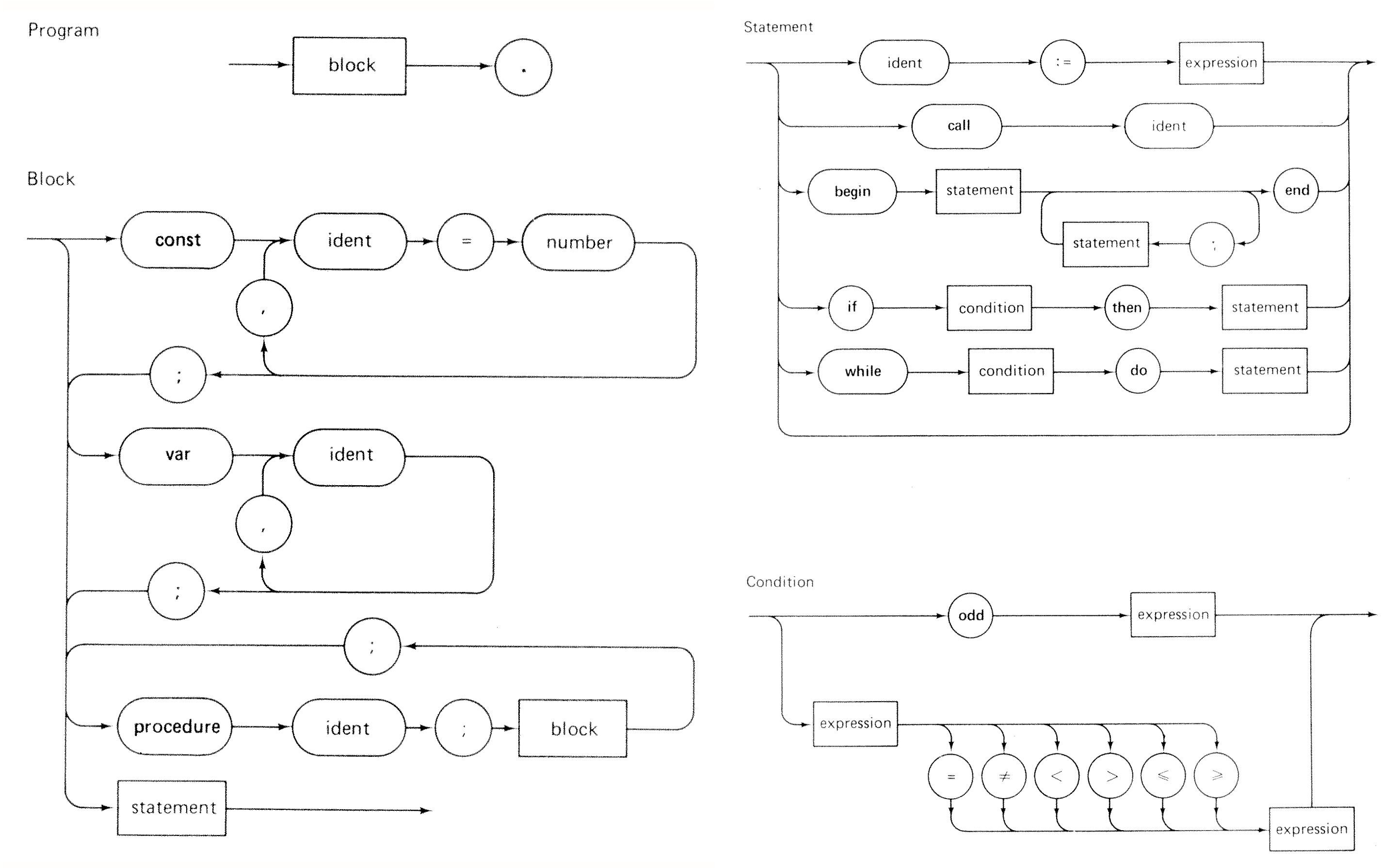
2. (20%)

   (a) (6%) Explain the difference between *type coercion* and *type casting*. Use an example if possible.

   (b) (8%) Languages such as `C++` and `Ada` support *generic* (or template) data types, i.e., parametric polymorphism, where a data structure or a function or a class/package can take on a *type* parameter, which can then be instantiated to different type at compile time. Explain this concept with a simple example in pseudo code.

   (c) (6%) What is *ad hoc* polymorphism?

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



(a) (8%) If A is a non-terminal symbol, then *first*(A) is the set of terminal symbols which begin as a terminal symbol in all derivations of A. Calculate the set *first*(`Block`).

(b) (12%) Assume the following procedures are given:

- `Statement()` which parses the inputs up to a `Statement`;
- `Accept( t :  Token )` which accepts the next input if it is a token `t` else it emits an error.

Sketch (in pseudo code) a *recursive decent* parser for the non-terminal `Block` specified by this syntax diagram.

4. (20%)

   (a) (8%) What is the purpose of an *activation record*?.

   (b) (6%) What are typically stored inside an *activation record*?

   (c) (6%) To implement the runtime stack of a language such as C, which doesn't have nested function scope (i.e., no function declaration inside another function), what is not needed inside the activation record and why?

5. (20%)

   (a) (5%) Pascal is an example of a *statically*-typed language, while LISP is an example of a *dynamically*-typed language. Explain the pros and cons of each approach.

   (b) (5%) Many programming languages today use both *name* and *structural* equivalences in type checking. Explain their differences.

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

```
<Type> ::=  "Int"   |   "Bool"   |   "array of" <Type>   | <Var>
            <Type> "-->" <Type>    |    <Type> "x"  <Type>
<Var>  ::=  a1  | a2  | ...
```

Calculate a *most general type* of the function `f` below. Show your reasoning and steps. (Note: It is not acceptable if you just provide an answer.)

```
function f  ( g : ???  ,  x : ???,  y : ??? ) return ??? {
   return g ( g(x,y), y );
}
```