CSc 330 F01

Test 1

NAME:	STUDENT NO:	
11A1V113	BIODENI NO.	

1. (20%) You are given:

- an Ada compiler, written in Ada, that translates Ada into a virtual machine P-code.
- an Ada interpreter implemented in P-code, and
- an x86 executable compiler for P-code that translates P-code into x86 machine code.

Show how you can compile a program **A** written in Ada into an executable program **A*** running on your host x86 machine. (Note: You cannot execute P-code on your host computer directly.)

2. (20%)

(a) (10%) Most imperative programming languages use *static binding* and *lexical scoping*. Define what they are. Does Ada use both?

(b) (10%) What is an *activation record*? Explain how it is used to provide runtime support for *scope* and *extent* of variables.

3. (20%) Given the following extended BNF grammar for a very small subset of Ada:

(a) (10%) Draw a syntax diagram for this subset of Ada.

(b) (10%)

Is the following a syntactically valid input for this subset? If so, show a derivation tree. If not, why not? Show a partial derivation tree where it fails. (Note: You may use acronyms, e.g., <S_S> for <simple_statement>, and <Ss> for <statements>, etc.)

loop if a = 5 then a := a + 1; null; else a := 2 * a; end if; end loop;

4. (20%)

(a) (15%) For each of the following type construction methods, give a simple type declaration in Ada that illustrates the concept. (**Note**: Your syntax *must* be a valid Ada declaration.)

i. (5%) sequence

ii. (5%) product

iii. (5%) sum

(b) (5%) Does Ada use name equivalence or structural equivalence? Illustrate with a simple Ada example.

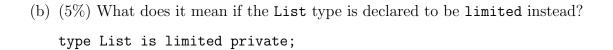
5. (20%) Use the following Doubly_Linked_List example in Ada. (It is a much simplified version of the online example; otherwise, it has the *same* operational meaning.)

```
package Doubly_Linked_List is
 type List is private;
 procedure Prepend (L : in out List; D : in Integer);
 procedure Append (L : in out List; D : in Integer);
 procedure Delete_All (L : in out List);
  function Is_Empty (L : in List) return Boolean;
  function "=" (Left : in List; Right : in List) return Boolean;
private
  type Data_Store;
  type Data_Store_Access is access Data_Store;
  type List_Head;
  type List_Head_Access is access List_Head;
 type Data_Store is record
     element : Integer;
     Next : Data_Store_Access;
     Previous : Data_Store_Access;
  end record;
 type List_Head is
 record
     First : Data_Store_Access;
     Last : Data_Store_Access;
  end record;
  type List is record
     Head : List_Head_Access;
  end record;
end Doubly_Linked_List;
```

(a) (10%) Explain in your own words why the following declarations are used, i.e., the package designer's intention.

```
i. (5\%) type List is private;
```

```
ii. (5\%) function "=" (Left : in List; Right : in List) return Boolean;
```



- (c) (5%) Write down a declaration for the following additional functions:
 - i. Member—for testing membership of an Integer in a List

ii. Delete—for removing an Integer in a List