1. Consider the following grammar:

\[
L \rightarrow L ; S | S \\
S \rightarrow id = E | id (E) \\
E \rightarrow id | num
\]

a) (10 marks) Construct the DFA of LR (0) items for this grammar

b) (5 marks) Construct the SLR(1) parsing table
2. Consider the following grammar for simple declarations:

\[ 
\text{decl-list} \rightarrow \text{decl-list} \ | \ \text{decl} \\
\text{decl} \rightarrow \text{simple-decl} \ | \ \text{structured-decl} \\
\text{simple-decl} \rightarrow \text{type} \ \text{var-list} \\
\text{structured-decl} \rightarrow \text{type} \ \text{var[\text{num}] } // \text{id[\text{num}]} \text{ means an array whose size is given by \text{num.val}} \\
\text{type} \rightarrow \text{integer} \ | \ \text{real} \\
\text{var-list} \rightarrow \text{var}, \ \text{var-list} \ | \ \text{var} \\
\text{var} \rightarrow \text{id} 
\]

a) (10 marks) Write an attribute grammar to insert the type of the variables in the symbol table. If it is an array you have to insert its elements type and its size in addition to indicating that this variable is a structured variable.

(Hint: consider the type field in the symbol table as a structure that has the fields:

- **category** (simple/structured),
- **element-type** which includes simply integer or real in case of a simple type, and includes the element type of an array (also integer or real) in case of an array, and
- **size** which includes the size of an array and null in case of simple types you can also assume that you have the following semantic actions: insert-category, insert-type, insert-size, and insert-link with appropriate parameters.)
b) (5 marks) Draw dependency graphs corresponding to each grammar rule for the attribute grammar of (a) and for the string `integer a,b; integer c[5]`
3. a) (7 marks) Show the stack with all activation record instances, including access and control links, when execution reaches position 1 in the following skeletal program.

```
Procedure BIGSUB;
  Procedure C;
    Procedure D;
      .......... \rightarrow 1
    End; {D}
  End; {C}
........
D;

End; {C}
Procedure A(f: boolean);
  Procedure B;
    ..........  
    A(false)
  End; {B}
..........
If f then B else C

End; {A}
............
A(true);
End; {BIGSUB}
```

The calling sequence for this program for execution to reach D is

- BIGSUB calls A
- A calls B
- B calls A
- A calls C
- C calls D
b) (5 marks) Write a sequence of intermediate code instructions that represents the semantic of a procedure call in a language that allows nested procedures.

c) (3 marks) Write a sequence of intermediate code instructions that represents the semantic of a return from a procedure.
4. Consider the following grammar for the assignment and for statements:

\[
\begin{align*}
\text{stmt} & \rightarrow \text{ass-stmt} \mid \text{for-stmt} \\
\text{ass-stmt} & \rightarrow \text{var} := \text{exp} \\
\text{for-stmt} & \rightarrow \text{for} \text{ var} = \text{exp} \text{ to } \text{exp} \text{ step } \text{exp} \text{ do stmt}
\end{align*}
\]

a) (10 marks) Write an attribute grammar for generating three-address code for the above grammar.
(Hint: the informal semantic of the for statement is to initialize the loop control variable, the final variable, and the step variable at the beginning by the expressions given, and then loop by incrementing the control variable by the step variable until the control variable value is greater than the value of the final variable).
b) (5 marks) Give the sequence of three-address code instructions corresponding to the following fragment using the above attribute grammar

```
for i = 1 to n step m+k do
  s := s+i
```