<Homework>

1. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position ① in the following skeletal program. Assume **Bigsub** is at level 1.

```
procedure Bigsub is
  procedure A is
    procedure B is
      begin -- of B
        . . . ①
      end; -- of B
    procedure C is
      begin -- of C
        . . .
        B;
        . . .
      end; -- of C
    begin -- of A
      . . .
      C;
      . . .
    end; -- of A
  begin -- of Bigsub
    . . .
    A;
    . . .
end; -- of Bigsub
```

2. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position ① in the following skeletal program. Assume **Bigsub** is at level 1.

```
procedure Bigsub is
  MySum : Float;
  procedure A is
    X : Integer;
    procedure B(Sum : Float) is
      Y, Z : Float;
      begin -- of B
        . . .
        C(Z)
        . . .
      end; -- of B
    begin -- of A
      . . .
      B(X);
      . . .
    end; -- of A
  procedure C(Plums : Float) is
    begin -- of C
      . . . ①
  end; -- of C
  L : Float;
  begin -- of Bigsub
    . . .
    A;
    . . .
  end; -- of Bigsub
```

```
procedure Bigsub is
  procedure A(Flag:Boolean) is
    procedure B is
      . . .
       A(false);
    end; -- of B
  begin -- of A
     if flag
        then B;
        else C;
     . . .
  end; -- of A
  procedure C is
    procedure D is
       . . . ①
    end; -- of D
    . . .
    D;
  end; -- of C
  begin -- of Bigsub
    . . .
    A(true);
    . . .
  end; -- of Bigsub
```

Bigsub calls A
A calls B
B calls A
A calls C
C calls D

3. Show the stack with all activation record instances, including static and dynamic chains, when execution reaches position ① in the following skeletal program. Assume Bigsub is at level 1.

4. Show the stack with all activation record instances, including dynamic chain, when execution reaches position ① in the following skeletal program. This program uses the deep-access method to implement dynamic scoping.

```
void fun1() {
  float a;
  . . .
}
void fun2() {
  int b, c;
  . . .
}
void fun3() {
  float d;
  . . . ①
}
void main() {
  char e, f, g;
  . . .
}

main calls fun2
fun2 calls fun1
fun1 calls fun1
fun1 calls fun3
```

5. Assume that the program of Problem 4 is implemented using the shallow-access method using a stack for each variable name. Show the stacks for the time of the execution of fun3, assuming execution found its way to that point through the sequence of calls shown in Problem 4.
6. Although local variables in Java methods are dynamically allocated at the beginning of each activation, under what circumstances could the value of a local variable in a particular activation retain the value of the previous activation?