17. What is a static ancestor of a subprogram? What is a dynamic ancestor of a subprogram?
18. What is a block?
19. What are the advantages and disadvantages of dynamic scoping?
20. What are the advantages of named constants?

PROBLEM SET

1. Decide which of the following identifier forms is most readable, and then support that decision.

SumOfSales
sum_of_sales
SUMOFSALES

2. Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?

3. Write a simple assignment statement with one arithmetic operator in some language you know. For each component of the statement, list the various bindings that are required to determine the semantics when the statement is executed. For each binding, indicate the binding time used for the language.

4. Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.

5. Describe a situation when a history-sensitive variable in a subprogram is useful.

6. Consider the following Ada skeletal program:

procedure Main is
    X : Integer;
procedure Sub3;  -- This is a declaration of Sub3
                 -- It allows Sub1 to call it
procedure Sub1 is
    X : Integer;
procedure Sub2 is
    begin  -- of Sub2
    ...
    end;  -- of Sub2
begin  -- of Sub1
    ...
    end;  -- of Sub1
procedure Sub3 is
    begin  -- of Sub3
    ...

end; -- of Sub3
begin -- of Main
...
end; -- of Main

Assume that the execution of this program is in the following unit order:

Main calls Sub1
Sub1 calls Sub2
Sub2 calls Sub3

a. Assuming static scoping, in the following which declaration of \( x \) is the correct one for a reference to \( x \)?

i. Sub1
ii. Sub2
iii. Sub3

b. Repeat part a, but assume dynamic scoping.

7. Assume the following Ada program was compiled and executed using static-scoping rules. What value of \( x \) is printed in procedure Sub1? Under dynamic-scoping rules, what value of \( x \) is printed in procedure Sub1?

```ada
procedure Main is
  X : Integer;
procedure Sub1 is
  begin -- of Sub1
    Put(X);
  end; -- of Sub1
procedure Sub2 is
  X : Integer;
  begin -- of Sub2
    X := 10;
    Sub1
  end; -- of Sub2
begin -- of Main
  X := 5;
  Sub2
end; -- of Main
```

8. Consider the following program:

```ada
procedure Main is
  X, Y, Z : Integer;
procedure Sub1 is
  A, Y, Z : Integer;
procedure Sub2 is
  A, B, Z : Integer;
begin -- of Sub2
```

... end; -- of Sub2
begin -- of Sub1
...
end; -- of Sub1

procedure Sub3 is
A, X, W : Integer;
begin -- of Sub3
...
end; -- of Sub3
begin -- of Main
...
end; -- of Main

List all the variables, along with the program units where they are declared, that are visible in the bodies of Sub1, Sub2, and Sub3, assuming static scoping is used.

9. Consider the following program:

procedure Main is
X, Y, Z : Integer;
procedure Sub1 is
A, Y, Z : Integer;
begin -- of Sub1
...
end; -- of Sub1

procedure Sub2 is
A, X, W : Integer;
procedure Sub3 is
A, B, Z : Integer;
begin -- of Sub3
...
end; -- of Sub3
begin -- of Sub2
...
end; -- of Sub2
begin -- of Main
...
end; -- of Main

List all the variables, along with the program units where they are declared, that are visible in the bodies of Sub1, Sub2, and Sub3, assuming static scoping is used.

10. Consider the following C program:

void fun(void) {
    int a, b, c; /* definition 1 */
... while (...) {
    int b, c, d; /* definition 2 */
    ... <------------------ 1
    while (...) {
        int c, d, e; /* definition 3 */
        ... <------------------ 2
    }<------------------ 3
    ...
    ... <------------------ 4
}

For each of the four marked points in this function, list each visible variable, along with the number of the definition statement that defines it.

11. Consider the following skeletal C program:

```c
void fun1(void); /* prototype */
void fun2(void); /* prototype */
void fun3(void); /* prototype */
void main() {
    int a, b, c;
    ...
}
void fun1(void) {
    int b, c, d;
    ...
}
void fun2(void) {
    int c, d, e;
    ...
}
void fun3(void) {
    int d, e, f;
    ...
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

a. main calls fun1; fun1 calls fun2; fun2 calls fun3.
b. main calls fun1; fun1 calls fun3.
c. main calls fun2; fun2 calls fun3; fun3 calls fun1.
d. main calls fun3; fun3 calls fun1.
e. main calls fun1; fun1 calls fun3; fun3 calls fun2.
f. main calls fun3; fun3 calls fun2; fun2 calls fun1.

12. Consider the following program:

```plaintext
procedure Main is
    X, Y, Z : Integer;
procedure Sub1 is
    A, Y, Z : Integer;
    begin    -- of Sub1
        ...
    end;    -- of Sub1
procedure Sub2 is
    A, B, Z : Integer;
    begin    -- of Sub2
        ...
    end;    -- of Sub2
procedure Sub3 is
    A, X, W : Integer;
    begin    -- of Sub3
        ...
    end;    -- of Sub3
    begin    -- of Main
        ...
    end;    -- of Main
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

a. Main calls Sub1; Sub1 calls Sub2; Sub2 calls Sub3.

b. Main calls Sub1; Sub1 calls Sub3.

c. Main calls Sub2; Sub2 calls Sub3; Sub3 calls Sub1.

d. Main calls Sub3; Sub3 calls Sub1.

e. Main calls Sub1; Sub1 calls Sub3; Sub3 calls Sub2.

f. Main calls Sub3; Sub3 calls Sub2; Sub2 calls Sub1.

---

**Programming Exercises**

1. Perl allows both static and a kind of dynamic scoping. Write a Perl program that uses both and clearly shows the difference in effect of the two. Explain clearly the difference between the dynamic scoping described in this chapter and that implemented in Perl.

2. Write a COMMON LISP program that clearly shows the difference between static and dynamic scoping.