13. Explain the primary uses of a methodology and notation for describing the semantics of programming languages.
14. Why can machine languages not be used to define statements in operational semantics?
15. Describe the two levels of uses of operational semantics.
16. In denotational semantics, what are the syntactic and semantic domains?
17. What is stored in the state of a program for denotational semantics?
18. Which semantics approach is most widely known?
19. What two things must be defined for each language entity in order to construct a denotational description of the language?
20. Which part of an inference rule is the antecedent?
21. What is a predicate transformer function?
22. What does partial correctness mean for a loop construct?
23. On what branch of mathematics is axiomatic semantics based?
24. On what branch of mathematics is denotational semantics based?
25. What is the problem with using a software pure interpreter for operational semantics?
26. Explain what the preconditions and postconditions of a given statement mean in axiomatic semantics.
27. Describe the approach of using axiomatic semantics to prove the correctness of a given program.
28. Describe the basic concept of denotational semantics.
29. In what fundamental way do operational semantics and denotational semantics differ?

PROBLEM SET

1. The two mathematical models of language description are generation and recognition. Describe how each can define the syntax of a programming language.
2. Write EBNF descriptions for the following:
   a. A Java class definition header statement
   b. A Java method call statement
   c. A C **`switch`** statement
   d. A C **`union`** definition
   e. C **`float`** literals
3. Rewrite the BNF of Example 3.4 to give + precedence over * and force + to be right associative.

4. Rewrite the BNF of Example 3.4 to add the ++ and -- unary operators of Java.
5. Write a BNF description of the Boolean expressions of Java, including the three operators `&&`, `||`, and `!` and the relational expressions.
6. Using the grammar in Example 3.2, show a parse tree and a leftmost derivation for each of the following statements:
   a. `A = A * (B + (C * A))`
   b. `B = C * (A * C + B)`
   c. `A = A * (B + (C))`
7. Using the grammar in Example 3.4, show a parse tree and a leftmost derivation for each of the following statements:
   a. `A = ( A + B ) * C`
   b. `A = B + C + A`
   c. `A = A * (B + C)`
   d. `A = B * (C * (A + B))`
8. Prove that the following grammar is ambiguous:
   <S> → <A>
   <A> → <A> + <A> | <id>
   <id> → a | b | c
9. Modify the grammar of Example 3.4 to add a unary minus operator that has higher precedence than either + or *.
10. Describe, in English, the language defined by the following grammar:
    <S> → <A> <B> <C>
    <A> → a <A> | a
    <B> → b <B> | b
    <C> → c <C> | c
11. Consider the following grammar:
    <S> → <A> a <B> b
    <A> → <A> b | b
    <B> → a <B> | a
    Which of the following sentences are in the language generated by this grammar?
    a. baab
    b. bbbab
    c. bbaaaaa
    d. bbaab

12. Consider the following grammar:

    <S> → a <S> c <B> | <A> | b

    <A> → c <A> | c

    <B> → d | <A>

    Which of the following sentences are in the language generated by this grammar?

    a. abcd

    b. acccbd

    c. acccbcc

    d. acd

    e. accc

13. Write a grammar for the language consisting of strings that have n copies of the letter a followed by the same number of copies of the letter b, where $n > 0$. For example, the strings ab, aaaabbbb, and aaaaaaaabbbbbbbb are in the language but a, abb, ba, and aaabb are not.
14. Draw parse trees for the sentences aabb and aaaabbbb, as derived from the grammar of problem 13.
15. Convert the BNF of Example 3.1 to EBNF.
16. Convert the BNF of Example 3.3 to EBNF.
17. Convert the following EBNF to BNF:

    S → A { bA }

    A → a [b]A

18. What is the difference between an intrinsic attribute and a nonintrinsic synthesized attribute?
19. Write an attribute grammar whose BNF basis is that of Example 3.6 in Section 3.4.5 but whose language rules are as follows: Data types cannot be mixed in expressions, but assignment statements need not have the same types on both sides of the assignment operator.
20. Write an attribute grammar whose base BNF is that of Example 3.2 and whose type rules are the same as for the assignment statement example of Section 3.4.5.
21. Using the virtual machine instructions given in Section 3.5.1.1, give an operational semantic definition of the following:

    a. Java **`do-while`**

    b. Ada **`for`**

    c. C++ **`if-then-else`**

    d. C **`for`**

    e. C **`switch`**

22. Write a denotational semantics mapping function for the following statements:
    a. Ada **for**
    b. Java **do-while**
    c. Java Boolean expressions
    d. Java **for**
    e. C **switch**
23. Compute the weakest precondition for each of the following assignment statements and postconditions:
    a. `a = 2 * (b - 1) - 1 {a > 0}`
    b. `b = (c + 10) / 3 {b > 6}`
    c. `a = a + 2 * b - 1 {a > 1}`
    d. `x = 2 * y + x - 1 {x > 11}`
24. Compute the weakest precondition for each of the following sequences of assignment statements and their postconditions:
    a.
    ```
    a = 2 * b + 1;
    b = a - 3
    {b < 0}
    ```
    b.
    ```
    a = 3 * (2 * b + a);
    b = 2 * a - 1
    {b > 5}
    ```
25. Compute the weakest precondition for each of the following selection constructs and their postconditions:
    a.
    ```
    if (a == b)
      b = 2 * a + 1
    else
      b = 2 * a;
    {b > 1}
    ```
    b.
    ```
    if (x < y)
      x = x + 1
    else
      x = 3 * x
    {x < 0}
    ```
    c.
    ```
    if (x > y)
      y = 2 * x + 1
    else
      y = 3 * x - 1;
    {y > 3}
    ```

26. Explain the four criteria for proving the correctness of a logical pretest loop construct of the form **while** B **do** S **end**
27. Prove that (n + 1) * ... * n = 1
28. Prove the following program is correct:

```
{n > 0}
count = n;
sum = 0;
while count <> 0 do
  sum = sum + count;
  count = count - 1;
end
{sum = 1 + 2 + ... + n}
```