

**Programming Languages and Compilers**  
**General Examination**  
**Spring, 1999**  
**Friday, March 5, 1999**

There are four questions from programming languages and four questions from compilers. Answer **ALL** of questions from the compiler section and answer **three** of the four questions from the programming language section. Please only answer three questions from the programming language section; otherwise, we will choose the first three for grading.

Compiler General  
Answer All Questions

1. Define the following terms:

Handle  
Sentential Form  
Regular Grammar  
Terminal Symbol  
Lexical Scan

2. Is it possible that a derivation (construction or syntax tree) exists for a string from a grammar, but a top down parser will reject the string? Explain your answer.

3. What is done during a two pass assembler?

4. Given the grammar:

$E ::= E + E$   
 $E ::= E * E$   
 $E ::= (E)$   
 $E ::= id$

Show how a bottom up parser might operate on  $id * id + id$ .

5. Remove left recursion from  $T ::= TB5:P$

Left factor the rule  $Q ::= AD:AF$

6. Briefly describe what is done during code generation.

## Programming Languages

1. a. Explain the problem associated with the use of pointers in the following C code.

```
int *x;
...
{
    int y = 1;
    x = &y;
}
...
```

- b. Describe two possible programming-language solutions to the above problem in a.  
c. Explain the problem associated with the use of pointers in the following C++ code.

```
int *t;
t = new int;
t = new int;
...
```

- d. Describe two possible programming-language solutions to the above problem in c.

2. Consider the following lines of pseudocode:

```
int f(x: integer);
{
    if (x < 3) then return ( 1 );
    else return ( f(x-1) + f(x-3) + 1 );
}
```

- a) What value is returned when  $f(5)$  is called?  
b) What is the algorithmic complexity of this function?  
c) Trace the runtime stack for this function when  $f(5)$  is called.  
d) What is the scope (define scope) of  $x$  in the function  $f$ .  
e) What are the different modes (types) of argument passing mechanisms for this function  $f$ ? What differences would it make if the argument  $x$  was passed by any one of these different mechanisms (consider the specific call  $f(5)$  if you wish to be specific)?

3. Write an attribute grammar for the following problem. Use only synthesized attributes:

A binary numeral is a sequence of binary digits followed by a period followed by another sequence of binary digits. (Assume at least one binary digit, which may be 0, for each sequence). Write an attribute grammar that defines the semantics of a binary numeral as the real number value associated with the numeral as expressed in base 10 notation.

4. Answer each of the following

- A. Assume you are designing a language, what type of semantics is likely to be the most useful to you? Explain your answer.
- B. Given the following program segment, what is the loop invariant?

```

{ n ≥ 0 }
i := n; f := 1;
while i > 0 do
    f := f * i; i := i - 1;
end while
{ f = n! }

```

- C. Given the following denotational semantics for integer arithmetic, show the computation of  $(2+3*4)$ .

#### Syntactic Domains

E: Expression  
 N: Number  
 D: Digit

$E ::= E_1 + E_2 \mid E_1 - E_2 \mid E_1 \cdot E_2$   
 $\mid ( E ) \mid N$

$N ::= ND \mid D$

$D ::= 0 \mid 1 \mid \dots \mid 9$

#### Semantic Domains

Domain  $v$ : Integer =  $\{ \dots, -2, -1, 0, 1, 2, \dots \}$

#### Operations

$+$  : Integer  $\times$  Integer  $\longrightarrow$  Integer  
 $-$  : Integer  $\times$  Integer  $\longrightarrow$  Integer  
 $\cdot$  : Integer  $\times$  Integer  $\longrightarrow$  Integer

#### Semantic Functions

$E$  : Expression  $\longrightarrow$  Integer

$E[[E_1 + E_2]] = E[[E_1]] + E[[E_2]]$   
 $E[[E_1 - E_2]] = E[[E_1]] - E[[E_2]]$   
 $E[[E_1 \cdot E_2]] = E[[E_1]] \cdot E[[E_2]]$   
 $E[[( E )]] = E[[E]]$   
 $E[[N]] = N[[N]]$

N: Number  $\longrightarrow$  Integer

$N[[ND]] = 10 \cdot N[[N]] + N[[D]]$   
 $N[[D]] = D[[D]]$

D: Digit  $\longrightarrow$  Integer

$D[[0]] = 0, D[[1]] = 1, \dots, D[[9]] = 9$