

Programming Languages and Compilers
General Examination
Spring, 2004
Friday, March 12 from 9:00 to 12:00

There are four questions from programming languages and seven 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 Examination

Answer all questions

- I. Define the following terms
 - a) Syntax directed translation
 - b) Interpreter
 - c) Phrase
 - d) Call by address
- II. Describe some applications of a stack in compiler construction.
- III. Write a BNF grammar that defines all labels, where a label is any letter of the alphabet followed by any sequence of letters and digits.
- IV. Give two ways an interpreter differs from a compiler.
- V. Give the rules for decomposing code into blocks for optimization. Give the rules for recognizing a loop.
- VI. Given the grammar

```
<sym> ::= <alp><str> | <alp><gam>
<str> ::= <str><alpha> | <alpha>
<alpha> ::= <alp> | <num>
<alpha> ::= A|B
<num> ::= 1|2
<gam> ::= C|E
```

make any necessary transformations to the grammar for a top down parser.

- VII. Find a formula for storing a two dimensional array backwards by row. That is A(10,10), A(10,9), A(10,8)...A(9,10), A(9,9)...

Programming Languages General Examination Answer 3 of the 4 questions

I. Answer each of the following:

a) Consider the following lines of pseudocode:

```
#include <stdio.h>
void main ()
{
  int a, b, c;
  void sa();
  {
    ...
    subb();
    ...
  }
  void subb();
  {
    int a, a, d;
    ...
    a = 2;
    ...
    subc();
    ...
  }
  void subc();
  {
    ...
    printf("%d0, a);
    ...
  }
  ...
  a = 0;
  ...
  suba();
  ...
}
```

What is printed under static scoping? Under dynamic scoping? Define these terms.

b) Consider the following pseudocode declarations:

```
type
  pascal = integer;
  pl1 = integer
var
  a: pascal;
  b: pl1;
```

Are a and b type compatible? Why or why not?

c) Differentiate between static, stack-dynamic, explicit heap-dynamic, and implicit heap-dynamic

variables in terms of binding times.

d) Differentiate between static, fixed stack-dynamic, stack-dynamic, and heap-dynamic arrays in terms of binding times.

e) What are the different modes (types) of argument passing mechanisms for programming languages? Briefly define each of them.

II. Answer each of the following:

a) Consider the following lines of pseudocode:

```
#include <stdio.h>
int h(int);
void main ()
{
    int w;
    scanf("%d", w);
    printf("%d %d 0, w, h(w));
}
int h(int w)
{
    if w = 0 then return(1);
    else if w < 0 then return (1 / h(-w));
    else return (2 * h(w-1));
}
```

Is the function h recursive? How do you know? If it is recursive, then explain how the run-time stack is used in the implementation of this code to implement the recursion.

b) What elements need be present for object-oriented programming?

c) Consider the pseudocode statement:

```
y = don + g(don);
```

Explain how side effects that might be present in the function g and the order of operations could affect the value of the variable don.

d) Explain how left, versus right, association could affect the evaluation of an expression such as $2^{**}3^{**}4$, where $**$ is the exponentiation operator. What would be printed out under each of these mechanisms?

III. Answer each of the following:

- a) Prove the following code segment using the given pre- and post- conditions: (Axiomatic specification attached)

```
{x = Vx and y = Vy}
temp := x;
x := y;
y := temp;
{x = Vy and y = Vx}
```

- b) In axiomatic specifications, what is a loop invariant? Why is it important for proving programs correct? Is there a similar concept in denotational semantics? If so, explain. Give a simple example of a loop and its invariant.
- c) Order the following semantics approaches from least abstract to most abstract: Axiomatic, Operational, Denotational.

IV. Answer each of the following:

- a) Describe the basic concept, including all of the key components, of denotational semantics.

b) How do operational and denotational semantics differ? Be specific.

c) Compare and contrast attribute grammars and BNF grammars.

d) Compare and contrast BNF grammars and EBNF grammars.

d) Give an example of a situation where one would choose to use a denotational semantic representation instead of just using a BNF representation.

Table 4.5 Axiomatic Definition of Mini-language Core

$\frac{\{P\} S \{Q\}, Q \Rightarrow R}{\{P\} S \{R\}}$	Consequence-1
$\frac{P \Rightarrow Q, \{Q\} S \{R\}}{\{P\} S \{R\}}$	Consequence-2
$\frac{\{P\} S \{Q\}, \{P'\} S \{Q'\}}{\{P \& P'\} S \{Q \& Q'\}}$	Conjunction
$\frac{\{P\} S \{Q\}, \{P'\} S \{Q'\}}{\{P \mid P'\} S \{Q \mid Q'\}}$	Disjunction
$\frac{\{P\} S_1 \{Q\}, \{Q\} S_2 \{R\}}{\{P\} S_1 ; S_2 \{R\}}$	Composition
$\frac{\{P\} \text{input } W_1 ; \text{input } W_2 \{Q\}}{\{P\} \text{input } W_1, W_2 \{Q\}}$	Input list
$\frac{\{P\} \text{output } W_1 ; \text{output } W_2 \{Q\}}{\{P\} \text{output } W_1, W_2 \{Q\}}$	Output list
$\{IN = \langle K \rangle L \& P[K/V]\} \text{input } V, \{IN = L \& P\}$	Input
$\{OUT = L \& P \& V = K\} \text{output } V, \{OUT = L \langle K \rangle \& P \& V = K\}$	Output
$\{P[E/V]\} V := E \{P\}$	Assignment
$\frac{\{P \& B\} S \{Q\}, P \& \neg B \Rightarrow Q}{\{P\} \text{if } B \text{ then } S \text{ end if } \{Q\}}$	If-then
$\frac{\{P \& B\} S_1 \{Q\}, \{P \& \neg B\} S_2 \{Q\}}{\{P\} \text{if } B \text{ then } S_1 \text{ else } S_2 \text{ end if } \{Q\}}$	If-then-else
$\frac{\{P \& B\} S \{P\}}{\{P\} \text{while } B \text{ loop } S \text{ end loop } \{P \& \neg B\}}$	Loop