King Fahd University of Petroleum and Minerals

Department of Information and Computer Science

ICS 410: Programming Languages

Spring 2006-2007 (062)

Date: 4-June-2007 **Final Exam** **Time Slot:** 7:00 p.m. – 9:00 p.m.

Location: 24-137 **Total Marks:** 300

Name:	**Student ID #:**

Notes:

- Check that you have **seven** (8) pages, including this one, containing **six** (6) questions.
- Please skim through all the questions, make sure that you understand them, and then attempt to answer them with a time-allocation in mind. If any question is not clear, get it clarified during the **first fifteen minutes.**
- If you need to make any reasonable assumptions, please state them clearly as part of your answers.
- There are six questions in this exam each focusing on one of the chapters. You are expected to answer all of them.

Scores:

Question	**Points**	**Score**
Q.1: Basic Principles of Programming Languages	**30**	
Q.2: Defining Syntax and Semantics of Programming Languages.	**50**	
Q. 3: Variables Scoping and Data Types	**30**	
Q. 4: Expressions and Control Structures	**50**	
Q. 5: Subprograms Design Concepts	**60**	
Q. 6: Subprogram Implementation	**80**	
Total	**300**	

Question 1 [Basic Principles of Programming Languages]: (30 points)

Answer **only t**wo the following parts:

i. What are the main characteristics of *Business* and *Artificial Intelligence* Programming languages?

ii. List three *cost factors* of a programming language and briefly explain one of them.

iii. Briefly explain how *data types and structures* of a programming langue may affect its readability. Given an example.

iv. *Intermediate code generation* and *code generation* are two major steps in the compilation of a program. Briefly explain them.

Question 2 [Defining Syntax and Semantics of Programming Languages]: (50 points)

A. Answer **only two** of the following parts: **(30)**

i. Briefly explain how a language *recognizer* works.

ii. What are the main *components* of a BNF grammar?

iii. Describe the basic concept of *axiomatic semantics*.

B. Consider the following grammar. For your convenience rules are given labels. In your answer you can refer to those rule numbers and also use the abbreviations: <a> for <assign>, <i> for <id>, <e> for <exp>:

```
R1:     <assign>     → <id> = <exp>
R2:     <exp>        → <id>
R3:     <exp>        → <id> + <exp>
R4:     <exp>        → <id> * <exp>
R5:     <exp>        → (<exp>)
R6:     <id>         → A | B | C
```

Show the ***right-most derivation*** of `A = A * (B + (C*A))` **(20)**

Question 3 [Variables Scoping and Data Types]: (30 points)

A. Answer **only one** of the following parts: **(15)**

i. What is the difference between *key-words* and *reserved-words* of a programming language? Give an example for each.

ii. Define *binding* and *binding time*. Give an example of a binding time.

B. Consider the following skeletal program: **(15)**

```
program Main;
  var D : integer;
  procedure sub1;
     var A, B, C : integer;
     procedure sub2(E);
       var B, E : integer;
     begin { sub2 }
       A := D + B + E;      <---------------------- test point
     end;   { sub2 }
  begin {sub1}
   sub2 (10);
  end; {sub1}
  begin { Main }
   sub1;
  end;  { Main }
```

Assuming that *static scoping* is used and given the following calling sequence, what variables are visible during execution of the statement shown as "test point". Include with each visible variable the name of the procedure in which it was defined: **`main calls sub1; sub1 calls sub2.`**

Question 4: (50 points)

A. Answer **only two** of the following parts: **(30)**

i. What is the purpose of using a *compound assignment operator* in a programming language? Give an example of one.

ii. Explain what actually happens when a C compiler evaluates the following expression: `a<b<c`?

iii. What are the common solutions to the *nesting problem* for two-way selectors?

iv. What is the main reason *user-located loop control* statements were invented? Give on design issue when implementing such constructs.

B. Consider the following C program. What are the values of the ***sum1 and sum2*** in each of the parts (i) and (ii) below? **(20)**

```
void fun(int *k) {
     *k += 4;;
     Return 3 * (*k) -1;
  }
}

void main() {
     int i= 10, j = 10, sum1, sum2;
     sum1 = (i/2) + fun (&i);
     sum2 = fun (&j) + (j/2);
}
```

i) Assuming the operands in the expression are evaluated left to right?

_____

_____

ii) Assuming the operands in the expression are evaluated right to left

_____

_____

Question 5: (60 points)

A. Answer **only two** of the following parts: **(30)**

i) What are the three *semantic models* of parameter passing? Briefly explain one of them.

ii) What is the difference between *generic* and *overloaded* subprograms?

iii) What are the general characteristics of subprograms**?**

iv) One of the design issues for a programming language that supports subprograms is to handle subprogram names that can be passed as parameters to other programs. Explain what is the major design issue here.

B. Consider the following C program. What are the values of the ***list*** array in each of the parts (i), (ii) and (iii) below? **(30)**

```
void fun(int first, int second) {
     first += first;
     second += second;
  }
}

void main() {
    int list[2] = {1,3};
    fun(list[1], list[0]);
}
```

ii) Assuming that parameters are passed *by value*.

_____

ii) Assuming that parameters are passed *by value-result*.

_____

iii) Assuming that parameters are passed *by reference*.

_____

Question 6: (80 points)

A. Answer **only two** of the following parts: **(30)**

i) List three of the tasks to be implemented when executing a **call** statement

ii) List two reasons why handling subprograms in programming languages that support stack-dynamic local variables is complex.

iii) Briefly define *static-depth* and *nesting-depth* of a subprogram in a language that support static scoping.

B. Consider the following program, written in a language that support static scoping:

```
program Main;
   var X : integer;
   procedure Test;
     var A, B, C : integer;
     procedure Multiply;
       var A, D : integer;
       begin { Multiply }
         A := B + X;  <---------------------- test
       end;  { Multiply }
     procedure Divide(X : integer);
         var B, E : integer;
       begin { Divide }
         multiply;
         A := D + E;
       end; { Divide }
   begin {Test}
     Divide(7);
   end; {Test}
   begin { Main }
     Test;
   end;  { Main }
```

i. Show the pair (*chain-offset*, *local-offset*) for each of the variables shown at the **test** point. **(15)**

ii. Using static chaining, draw the run-time stack (growing upwards), for the above program when execution reaches the test point. Include in your drawing the static links (labeled SL), dynamic links (labeled DL), arrows showing where each link points, variable names (labeled with the variable name), and variable values (including parameters). Use a heavy line between activation records. Be sure to label each activation record with the name of the procedure it is for, and clearly indicate to which activation record each static or dynamic link points. **{***If you need more space then use the facing page to show your solution***}.**? **(35)**