PRESIDENCY UNIVERSITY, BENGALURU
SCHOOL OF ENGINEERING

Max Marks: 40 | Max Time: 60 Mins | Weightage: 20 %

TEST 2 (OPEN BOOK)

I Semester AY 2017-2018 Course: CSE **214 Principles of Programming Languages** 28 OCT 2017

Instructions:

i. Write legibly
ii. Only Text Book, Reference Book and Class note book is allowed.
iii. Photo copy materials and loose sheets are not allowed.

Part A

(1 Q x 6 M= 6 Marks)

1. Explain the working of the following program if the parameters are (i) passed by value and (i) passed by name?

```
fun(int x, int y)
 {
     print( x/y); //print statement
 }
main( )
 {
     fun(5+8,2+1);  //function call
 }
```

Part B

(2 Q x 10 M= 20 Marks)

2. What is the output of the following program, if the parameter is (i) Passed by Value (ii) Passed by Value result and (iii) Passed by reference?

```
int n;
int fun(int p)
{
 n := n+1;
 p := p+n;
 print(n); //print statement
}

main( )
{
n := 5;
fun(n); // function call
print(n); //print statement
}
```

3. "Aliasing of parameters will lead to unexpected results when parameters are passed by reference". Illustrate with an example.

Part C

(1Q x 14 M= 14 Marks)

4. Consider the grammar of a Programming Language: The non terminals are enclosed within <NT> brackets, others are terminals, and <Program > is the starting symbol.
   The Productions rules are given below.

1) <*Program*> →<*functions*>comma<*functionbody*>comma
2) <*functions*>→<*function*><*functions*>/ ε
3) <*function*> →<*funsignature*><*functionbody*>
4) <*funsignature*>→ <*type*>< id> (<*params*>)>
5) <*type*>→ int / float/string
6) <*params*>→ <*type*>< id> comma <*params*> / ε
7) <*functionbody*>→{ <*declarations*> <*statements*> *<*E*>; }
8) <*declarations*>→ <*type*><id> ; <declarations >/ ε
9) <*statements*> →< id> =<*E*>;<statements>/ <id>=<id><more>;<statements>/ε
10) <*E*>→ <*E*>+<*E* >/ <*E*>*<*E*>/<*id*>/<*intigerliteral*>/<*floatliteral*>/<*stringliteral*>
11) <*more*> →(<*args*>)
12) <*args*> →< id> comma <*args*> / ε
13) <id>→ [a-z]$^{+}$ [A-Z] [0-9]*[a-z]*[A-Z]*

   a. Rewrite suitable production rule /s of the grammar to add the syntax for declaration of an array. The array may be single dimensional, two dimensional or any multidimensional. The example of integer array declaration is as shown below.

      Single dimensional array: int arrayName[10];
      Two dimensional array: int arrayName[10:10];
      ……..
      n dimensional array: int arrayName[10:10:…….:10];

      Note: int is a data type, **arrayName** is an identifier, and **10** is an integer literal.

   b. Derive the following program to show the correctness of your grammar.

```
,
{
int a[10:10];
*z;
}
,
```

----------END----------

PRESIDENCY UNIVERSITY, BENGALURU
SCHOOL OF ENGINEERING

Max Marks: 40 Max Time: 60 Mins Weightage: 20 %

TEST 1(OPEN BOOK)

I Semester 2017-2018 Course: CSE **214 Principles of Programming Languages** 23 SEPT 2017

Instructions:

i. Write legibly
ii. Scientific and non programmable calculators are permitted
iii. Only Text Book, Reference Book and Class note book is allowed.

Part A

(3Q x 4 M= 12 Marks)

1. The symbols {<, >, and < >} are used as logical operators {**Less than, Greater than and Not equal**} respectively in a new programming language. Discuss Programming language designer responsibility and Compile writer responsibility with respects to these symbols.
   Hint: Think in terms of operation binding and typing.

2. The symbols **{<, >, and < >}** are used as logical operators **{Less than, Greater than and Not equal}** respectively in a new programming language. Discuss Readability and Orthogonality w.r.t these symbols as a design choice in a programming language.

3. The symbols < > are used as logical operator **Not equal** in a new programming language. In C language, the symbols != are used as a logical operator **Not equal.** Discuss the effect in compiler writing or implementation stage with example.

Part B

(2 Q x 8 M= 16 Marks)

Consider the program fragment to answer the question number 4 and 5.

Void fun3(int min) { min=(min<c)?min:c; ***printf("%d", min);*** }	**Void fun2( )** { int b=15; int min; min= (a<b)?a :b; *fun3(min);* }	**void fun1( )** { int b; a=2; b=1; c=1; *fun2( );* ***printf("%d", min);*** }	int min =0,a=9,c=8; **main( )** { int min=20; int a=15,b=13,c=22; *fun1( );* ***printf("%d", min);*** }

4. Give the run time stack when the fun3 is executing. Mention clearly the dynamic link, frame pointer and stack pointer.
5. a) What is the output of the program when static scoping is used?
   b) What is the output of the program when dynamic scoping is used?

Part C

(1 Q x 12 M= 12 Marks)

6. Consider the grammar of a Programming Language: The non terminals are enclosed within <NT> brackets, others are terminals, and <Program > is the starting symbol.
   The Productions rules are given below.

1) <*Program*> →<*functions*>comma<*functionbody*>comma
2) <*functions*>→<*function*><*functions*>/ ε
3) <*function*> →<*funsignature*><*functionbody*>
4) <*funsignature*>→ <*type*>< id> (<*params*)>
5) <*type*>→ int / float/string
6) <*params*>→ <*type*>< id> comma <*params*> / ε
7) <*functionbody*>→{ <*declarations*> <*statements*> *<*E*>; }
8) <*declarations*>→ <*type*>><id> ; <declarations >/ ε
9) <*statements*> →< id> =<*E*>;<statements>/ <id>=<id><more>;<statements>/ε
10) <*E*>→ <*E*>+<*E* >/ <*E*>*<*E*>/<*id*>/<*intigerliteral*>/<*floatliteral*>/<*stringliteral*>
11) <*more*> →(<*args*>)
12) <*args*> →< id> comma <*args*> / ε
13) <id>→ [a-z]$^{+}$ [A-Z] [0-9]*[a-z]*[A-Z]*

a) Consider the following program. Prove that the program belongs to the grammar.
   Hint: Show the derivation.

```
comma
{
int a;
a = fun1(a,);
*a;
}
comma
```

b) Consider the following program. Prove that the program belongs to the grammar.
   Hint: Show the derivation.

```
comma
{
int a;
a = fun1(a);
*z;
}
comma
```

c) Identify the production rules with left factoring in the given grammar and eliminate it.
d) Identify a production rules with left recursion and eliminate it.

----------END----------