BITS Pilani, Dubai Campus
Dubai International Academic City, Dubai

III Year CS
Second Semester, 2011-2012

No of Questions
PART A: 5
PART B: 4
No of Pages: 3

Comprehensive Examination (CLOSED BOOK)

Course No: CS C362 **Course Title: Prog. Lang & Complier Const.**
Date: 12th Jun 2012 **Weightage: 40%**
Duration: 3 Hours **Max. Marks. 40**

(Answer PART –A and PART –B in separate answer books).

PART – A

1. a) Distinguish between EARLY Binding and LATE Binding, with examples. **[2M]**
   b)Explain the difference between CALL by VALUE and CALL by REFERENCE in parameter passing. **[2M]**
2. Which provides better security for program execution i) Virtual Machine Execution environment ii) Complied Program Execution environment? Justify **[3M]**
3. a) Given algorithm for construction of Predictive Parsing Table **[3M]**
   b) Explain with an example what is meant by left factoring of a grammar **[3M]**
4. Explain the working of the Java code fragment **[3M]**

```
class Body
{
    String name;
    double mass, volume;

    public Body( String n, double m, double v)
    {
        name = n;
        mass = m;
        volume = v;
    }

    public void display()
    {
        System.out.print(name + " : " + mass + " : " + volume);
    }
}
```

5. The predictive parsing table for the expression grammar is shown below

$$
\begin{aligned}
E &\rightarrow T\ E' \\
E' &\rightarrow +\ T\ E' \mid \epsilon \\
T &\rightarrow F\ T' \\
T' &\rightarrow *\ F\ T' \mid \epsilon \\
F &\rightarrow (\ E\ ) \mid \mathbf{id}
\end{aligned}
$$

NON - TERMINAL	INPUT SYMBOL					
	id	+	*	(	)	$
E	$E \to TE'$			$E \to TE'$		
E'		$E' \to +TE'$			$E' \to \epsilon$	$E' \to \epsilon$
T	$T \to FT'$			$T \to FT'$		
T'		$T' \to \epsilon$	$T' \to *FT'$		$T' \to \epsilon$	$T' \to \epsilon$
F	$F \to$ id			$F \to (E)$		

Show the moves made by the predictive parser in the input sentence **id * id + id** **[3M]**

PART B

1. For the 'C' code fragment shown give the three address code. Assume size of int is 4 bytes, and that array `a[]` and `n` are global variables. **[5M]**

```
void sort()
{
   int i, j, limit;
   int temp;
   limit = n-1;
   do {
        for (i=0;i<limit;i++){
             if (a[i] > a[i+1]) {
                  temp = a[i];
                  a[i] = a[i+1];
                  a[i+1] = temp;
             }
        }
        limit--;
   } while(limit>0);
}
```

2. For the expression shown **[2M+3M]**

((x + y) ↑ (x - y)) + ((x + y) ↑ x ↑ y)

a) Show the Directed Acyclic Graph (DAG).

b) Using the Syntax-Directed definition shown below, show the steps for the construction of DAG in a)

	PRODUCTION	SEMANTIC RULES
1)	$E \to E_1 + T$	$E.node =$ **new** $Node('+', E_1.node, T.node)$
2)	$E \to E_1 - T$	$E.node =$ **new** $Node('-', E_1.node, T.node)$
3)	$E \to T$	$E.node = T.node$
4)	$T \to (E)$	$T.node = E.node$
5)	$T \to$ **id**	$T.node =$ **new** $Leaf($**id**, **id**.$entry)$
6)	$T \to$ **num**	$T.node =$ **new** $Leaf($**num**, **num**.$val)$

3. **a)** Partition the three address code into Basic Blocks **[4M]**

1) f = -1
2) l = 0
3) h = n -1
4) if l > h goto 17
5) t1 = l + h
6) t2 = t1 / 2
7) t3 = t2 * 4
8) t4 = a[t3]
9) if t4 ≠ val goto 12
10) f = t2
11) goto 17
12) if t4 < val goto 15
13) h = t2 – 1
14) goto 4
15) l = t2 + 1
16) goto 4
17) EXIT

b) For the simple assignment given below show the code generated along with information in register descriptors and address descriptors. **[4M]**

e = (a + b) + (a – c) + (b + d)

Assume there are 4 registers R1, R2, and R3, R4 and that the machine instructions are of the form.

LD reg, mem : Load reg with contents of memory

ST reg, mem : Store contents of reg to memory

OP reg1, reg2, reg3 : reg1 = reg2 OP reg3, where OP can be ADD/SUB

4. a) For the program shown below **[2M]**

```
int x =10;
main()
{
    int y = 5;
    func1(x, y);
}
```

```
void func1(int b, int c) {
  int i;
  for (i = 0; i < b; i++) {
    func2(c);
  }
}
```

```
void func2(int a) {
 int temp1, temp2;
 /*do some thing */
}
```

Show progressively the stack frames (activation records), when

a) main() is just activated
b) func1() is just activated
c) func2() is just activated when i =1 in func1.
d) func2() has just finished execution and control has come back to main().

c) What is peephole optimization? Using simple examples explain **[2M]**

a) Reduntant Instruction Elimination
b) Flow-of-contol Optimization

♣ ♣ ♣ ♣ ♣

BITS Pilani, Dubai Campus
Dubai International Academic City, Dubai

III Year CS
Second Semester, 2011-2012

Test 2 (Open Book)

No of questions : 4
No of Pages : 2

Course No: CS C362 **Course Title: Prog. Lang & Complier Const.**
Date: 13th May 2012 **Weightage: 20%**
Duration: 50 minutes **Max. Marks. 20**

(Convention for grammar symbols is as described in Sec 4.2.2)

1. **a)** Consider the grammar $E \rightarrow E + E \mid E - E \mid E * E \mid E / E \mid (E) \mid$ **id** . Show that the grammar is ambiguous for the input sentence **id – id – id + id,** using parse trees **[2M]**

   b) Consider the grammar to parse Boolean expressions
   $A \rightarrow A$ **or** $B \mid B$
   $B \rightarrow B$ **and** $C \mid C$
   C→**not** $A \mid (A) \mid$ **id**
   Show the grammar after elimination of Left Recursion (if any). **[3M]**

2. For the grammar shown the FIRST, FOLLOW are given below.
   (1) *re* → *ae rop ae*
   (2) *ae* → *at ae'*
   (3) *ae'* → + *at ae'*
   (4) *ae'* → ε
   (5) *at* → *af at'*
   (6) *at'* → * *af at'*
   (7) *at'* → ε
   (8) *af* → - *af*
   (9) *af* → (*ae*)
   (10) *af* → **num**
   (11) *rop* → =
   (12) *rop* → >
   (13) *rop* → <

   First (*re*) = First (*ae*) = First (*at*) = First (*af*) = { -, (, **num** }
   First(ae') = {+, ε } First(at') = {*, ε }
   Follow(*re*) = { $ }
   Follow(*ae*) = Follow(ae') = { =, <, >,), $ }
   Follow(*at*) = Follow(at') = { +, =, <, >,), $ }
   Follow(*af*) = { *, +, =, <, >,), $ }
   Follow(*ro*) = { -, (, num }

P.T.O

Construct the Predictive Parsing Table. (Parsing table entries may be shown using the production num instead of the production itself) **[5M]**

Non Terminal	Input Symbols									
	num	+	-	*	(	)	=	<	>	$

3. a) For the augmented grammar shown below

(0) $E' \rightarrow E$
(1) $E \rightarrow E + T$
(2) $E \rightarrow T$
(3) $T \rightarrow T * F$
(4) $T \rightarrow F$
(5) $F \rightarrow (E)$
(6) $F \rightarrow$ **id**

Show how **i)** I_1 = CLOSURE(I) where $I = \{ E \rightarrow E + . T\}$, **ii)** GOTO ($I_1$, F) are computed for an LR(0) parser. **[2M]**

b) For the grammar shown in 5 a), the LR parsing table is given below **[3M]**

STATE	ACTION						GOTO		
	id	+	*	(	)	$	E	T	F
0	s5			s4			1	2	3
1		s6				acc			
2		r2	s7		r2	r2			
3		r4	r4		r4	r4			
4	s5			s4			8	2	3
5		r6	r6		r6	r6			
6	s5			s4				9	3
7	s5			s4					10
8		s6			s11				
9		r1	s7		r1	r1			
10		r3	r3		r3	r3			
11		r5	r5		r5	r5			

Figure 4.37: Parsing table for expression grammar

Show the moves made by an LR(0) parser for the input sentence **id * id * id$,** in the table form given below

Sl No	Stack	Symbols	Input	Action
1				

4. Consider the following syntax directed definition for a desk calculator program (Note: **n** represents newline)

Construct an *annotated* PARSE TREE for the following input expression:

-(3 + 2) * (4 + 1) **n** **[5M]**

****

PRODUCTION	SEMANTIC RULES
$L \rightarrow E$ **n**	print(E.val)
$E \rightarrow E_1 + T$	E.val = E_1.val + T.val
$E \rightarrow T$	E.val=T.val
$T \rightarrow T_1 * F$	T.val=T_1.val * F.val
$T \rightarrow F$	T.val=F.val
$F \rightarrow (E)$	F.val=E.val
$F \rightarrow -E$	F.val=-E.val
$F \rightarrow$ **digit**	F.val=**digit**.lexval

BITS Pilani, Dubai Campus
Dubai International Academic City, Dubai

III Year CS
Second Semester, 2011-2012

Test 1 (Closed Book)

No of questions : 5
No of Pages : 2

Course No: CS C362 | **Course Title: Prog. Lang & Complier Const.**
Date: 22th Mar 2012 | **Weightage: 25%**
Duration: 50 minutes | **Max. Marks. 25**

---

1. **a)** What is meant by Delayed Linking? **[2M]**
   b) Explain the advantages of Delayed Linking? **[3M]**

2. It is required to store the information regarding a **Course** with the following details **i)** c_code (type string), **ii)** c_name (type string) **iii)** c_credits (type integer).
   It has 2 constructors
   i) with no parameters
   ii) with 3 parameters (code, name, credits)
   It has a method **getCredits** which returns the credits of the course.
   (All the methods should be accessible outside the class)
   In Java, show
   i) Class definition of **Course**
   ii) How an **instance** of the class can be created with the following details
   c_code = "CS C362", c_name ="PLCC", c_credits = 3. **[5M]**

3. **a)** What is inheritance in Object Oriented Paradigm?
   How is it achieved in Java? **[2M]**

   b) Show the following relationship can be represented in Java.
   There is a base class called **Book.** There are 2 derived classes **ReferenceBook** and **LendingBook** **[3M]**

4. **a)** For the expression
   value = (a + b) * (c – d), where the data type of identifiers **value**, **a**, **c** is **float** and that of **b** and **d** is **integer** respectively,
   show the output of the following analysis phases of a typical compiler.
   i) lexical, **ii)** syntax and **iii)** semantic **[3M]**

   b) What is need of the code optimization phase of a compiler? **[2M]**

P.T.O

5. Write the output of the following 'C' code. **[5M]**

```
#include <stdio.h>
main ()
{
  void e (int *xx, int *nn);
  int x[10], i;
  int n = 1;
  for (i = 0; i < 10; i += 2)
    {
      x[i] = 2 * n;
      e (&x[i], &n);
      n++;
    }
}
void
e (int *xx, int *nn)
{
  int m, z;
  m = *nn * 2;
  z = *xx + 2;

  printf (" m = %d z= %d \n", m, z);
}
```

BITS PILANI, DUBAI CAMPUS
SECOND SEMESTER 2011 – 2012
THIRD YEAR (CS)
QUIZ 1

SET A

No of Questions: 10
No of Pages : 2

Course Code: CS C362
Course Title: Programming Languages and Compiler Construction
Duration: 20 minutes

Date: 06.03.12
Max Marks: 08
Weightage: 8%

Name: ..... **ID No:** ..... **Sec / Prog:** .....

Instructions: Write your answers in the blank space provided after each question. You may use the reverse side if necessary.

1. What do you understand by Programming Language Paradigms? **[1M]**

2. _____ programming paradigm facilitates computation by means of state changes. **[0.5M]**

3. Give 2 characteristics of declarative programming paradigms **[1M]**

4. Give 2 features of object oriented systems. **[1M]**

5. Functional Programming languages allow us to consider data as functions. Justify **[1M]**

6. What are predicates in Prolog like language? **[0.5M]**

7. Write the interpretation in English sentence (if..then/Query) of the following prolog statements. **[1M]**

   i) tedge(Node1,Node2) :- edge(Node1, SomeNode),edge(SomeNode, Node2).

   ii) ?- parent (X, sally).

8. Interpreters are usually slower than Compilers, Why? **[1M]**

9. Explain how Virtual Machines achieves cross-platform execution of programs **[0.5M]**

10. For the following pico lisp program. **[0.5M]**

```
(de func (n)
  (if (= n 1)
     1
     (+ n (func (- n 1) ) ) ) )
```

What is the output of `(func 5)` ?

♣ ♣ ♣ ♣ ♣

BITS PILANI, DUBAI CAMPUS
SECOND SEMESTER 2011 – 2012
THIRD YEAR (CS)
QUIZ 2

SET A
No of Questions: 8
No of Pages : 2

Course Code: CS C362
Course Title: Programming Languages and Compiler Construction
Duration: 20 minutes

Date: 24.04.12
Max Marks: 7
Weightage: 7%

Name: ..... **ID No:** ..... **Sec / Prog:** .....

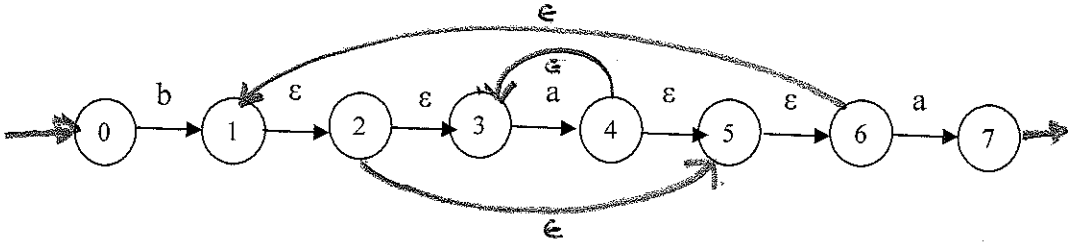
Instructions: Write your answers in the blank space provided after each question. You may use the reverse side if necessary.

1. Distinguish between ***pattern*** and ***lexeme*** in the context of lexical anlalysis? **[0.5M]**

2. Give the regular expression for the following language defined over the alphabet {**a**,**b**}**[0.5M]**
   a followed by at least one **a** or **b** followed by **a**

3. Draw the NFA for the regular expression **a(a|b)*b**, define over the alphabet **{a,b}** **[1M]**

4. For the following NFA, compute the e-closure of the NFA state **6** **[1M]**



5. Give the transition diagram for recognizing the following tokens +, -, +=, -= **[1M]**

6. Give the conflict resolution rules in Lex? **[1M]**

7. Describe in words the set of strings recognized by the following lex regular expressions **[1M]**
   i) ^[a-e]

   ii) [a-e]$

8. Given the lex specification write the output for each of the inputs shown **[1M]**
   Lex spec. %%
   abb { printf ("1");
   abab { printf ("2");

   i) Input: abba

   Output:

   ii) Input: abababbbabb

   Output:

♣ ♣ ♣ ♣ ♣