Registration No. : □□□□□□□□□□

Total number of printed pages – 4

B. Tech

PCCS 4305

Sixth Semester Regular Examination – 2015

COMPILER DESIGN

BRANCH : CSE

QUESTION CODE : J 282

Full Marks – 70

Time : 3 Hours

Answer Question No. 1 which is compulsory and any five from the rest.

The figures in the right-hand margin indicate marks.

1. Answer the following questions : 2 ×10

(a) What is front end and back end in compilation process ?

(b) What is the intermediate code representation for the expression a or b and not c ?

(c) Eliminate left recursion from the grammar.

$S \rightarrow Aa \mid b$

$A \rightarrow Ac \mid Sd \mid e$

(d) What is meant by register descriptors and address descriptors ?

(e) Assume that the SLR parser for a grammar G has *n*1 states and the LALR parser for G has *n*2 states. Then find the relationship between *n*1 and *n*2.

(f) Do the necessary changes to the grammar given below to make it suitable for LL (1) parser.

$S \rightarrow (L) \mid a$

$L \rightarrow L, S \mid S$

(g) Define code optimization and optimizing compiler.

(h) Give the syntax-directed definition for ***if-else*** statement.

(i) How do you calculate the cost of an instruction ?

(j) Suggest a suitable approach for computing hash function.

P.T.O.

2. (a) For the R.E. (a/b)*a(a/b). Draw the NFA. Obtain DFA form NFA. Minimize DFA using Π_{new} construction. Write down the algorithm wherever necessary. 5

(b) Develop a LEX specification to generate a scanner that will take a decimal number between 1 to 999 in words and prints its value as output. For example :

Input : one hundred ten

Output would be : 110 5

3. (a) What is an operator precedence grammar ? Prove that the following grammar is not operator precedence with assumptions of precedence relations. Transfer the grammar into an equivalent grammar that is both operator precedence and unambiguous :

E -> E – E | E/E |(E)|id

Calculate the operator precedence functions if exist for all the operators present in the grammar. 5

(b) Construct an LALR(1) parsing table for the following grammar : 5

S → AA

A → aA

A → b

4. Construct a table-based LL(1) predictive parser for the following grammar G = ({bexpr, bterm, bfactor}, {not, or, and, (,), true, false}, {P}, {bexpr}) with P given below :

bexpr → bexpr or bterm | bterm

bterm → bterm and bfactor | bfactor

bfactor → not bfactor | (bexpr) | true | false

For this grammar answer the following questions:

(a) Remove left recursion from G.

(b) Left factor the resulting grammar in (a).

(c) Computer the FIRST and FOLLOW sets for the non-terminals.

(d) Construct the LL parsing table.

(e) Verify your construction by showing the parse tree for the input string "true or not (true and false)" 10

5. (a) Consider the program fragment:

```
sum=0;
for(i=1;i<=20;i++)
sum=sum + a[i]+b[i];
```

And generate the three address code for it assuming that there are four bytes per word. 5

(b) Discuss the syntax directed translation schemes to specify the translation of Boolean expression containing AND, NOT and OR. 5

6. (a) Discuss in detail about error recovery in LR parsing for the following Grammar

E → E+E
E → E*E
E → id

Describe the behavior of the parser for the input id +*id by constructing the SLR parser, parsing table without parsing action conflicts and the parsing table with error routines. 5

(b) Consider the following C program. Eliminate the dead code from that program. 5

```
int foo(void)
{
int a = 24;
int b = 25;
int c;
c = a << 2;
return c;
b = 24;
return 0;
}
```

7. (a) What data is stored in the symbol table for a number token ? Explain in detail using symbol table management techniques. 5

(b) For the program below:

i. Determine the corresponding control flow graph.

ii. Determine the dominators of each node in the CFG

iii. Show the immediate dominator tree

iv. Identify the set of nodes in each natural loop. Are the loops nested ? Why ?

```
i = 0;
while (i < N) do
{
for(i=0; j < i; j++)
{
if(a[i] < a[j])
{
tmp = a[j];
a[j] = a[i]
a[i] = tmp; }
else
if(a[i] == a[j]) continue;
else break;
}
i = i + 1;
}
```

Assume there are entry and exit nodes corresponding to the entry point and exit points of the code. 5

8. Write short notes on any **two** of the following : 5×2

(a) peephole optimization

(b) activation records

(c) symbol tables

(d) dependency graphs.