Ref. No. Ex/T/CSE/T/323A/2017

BCSE 3RD YEAR EXAMINATION 2017

(2nd Semester)

Compiler Design

Time: Three Hours Full Marks 100

Answer question no. 1 and any four from the rest

1. Answer any five questions. 20
   a) Explain how you implement three-address code.
   b) What is *Predictive Parsing*?
   c) With examples explain the difference between "*useless code elimination*" and "*unreachable code elimination*".
   d) How can you check whether a grammar is LL(1) or not?
   e) Define the terms: token, pattern and lexeme with examples.
   f) Write at least four strings (each with at least four symbols) generated from the following grammar: $S \rightarrow a\,S\,b\,S \mid b\,S\,a\,S \mid e$
   h) What is a viable prefix? Write four viable prefixes for the string "id+id*id" using the grammar:
   E -> E + T | T
   T -> T * F | F
   F -> id

2. (a) Define context sensitive grammar and explain its difference with context free grammar with suitable examples.
   (b) Define regular grammar. How can a language based on regular grammar be recognized?
   Write regular expressions to define
   i) Floating point numbers upto two digits after decimal point.
   ii) Strings containing the characters {$, c, ., 0, ... 9} that are legal descriptions of monetary quantities. For example, $33, $5.32, 45c etc.
   iii) Strings containing any consonants and only 'a' and 'u' as vowels.
   (c) Convert the regular expression $(p|q)^*sr|t$ into an DFA.
   4+6+10=20

3. (a) What are the implications of computing First and Follow sets in the context of top-down parsing?
   (b) Consider the grammar:
   $S \rightarrow ABC$
   $A \rightarrow aA \mid \varepsilon$
   $B \rightarrow bB \mid \varepsilon$
   $C \rightarrow c$
   (Terminals = {a, b, c}, Non-terminals = {S, A, B}, Start Symbol = S)
   Construct the LL(1) parsing table for the grammar.
   (c) Draw a parse tree for the string "aaabbc" using the above grammar. Also show the leftmost and rightmost derivations of the string.
   4+8+8=20

4. Consider the following grammar:

S --> AB
S --> BA
A --> aaB
A --> a
B --> bbA
B --> a

(Terminals = {a, b}, Non-terminals = {S, A, B}, Start Symbol = S)
(a) Construct LR(0) item set for the above grammar.
(b) Construct the SLR parsing table for the above grammar. Is the grammar SLR? Justify your answer.
(c) Show the actions of the parser for the input string: (aaabba).

8+8+4=20

5. Consider the following grammar:

S --> id | V := E
V --> id
E --> V | n

(a) Construct LR(0) item set for the above grammar.
(b) Construct the SLR parsing table for the above grammar.
(c) What kind of conflict do you find in the SLR parsing table? Explain why.
(d) Construct LR(1) item set. Explain how the conflicts are resolved using LR(1) item set.

5+5+3+7=20

6. (a) During which phases of a compiler a symbol table is modified?
(b) What are the two main operations performed on a symbol table? Which implementation of symbol table would have O(log n) complexity per operation? Describe with an example.
(c) How do you store a variable length identifier in a symbol table? How can you handle multiple declarations of an identifier in a program? Discuss the implementation of a symbol table in such circumstances.
(d) Consider the grammar:

decl --> type var-list
type --> int | float
var-list --> id , var-list | id

Using attribute grammar, draw an annotated parse tree for the statement ***float*** ***x, y, z*** and demonstrate how data types are associated with the variables ***x, y*** and ***z***.

2+5+6+3+6=20

7. (a) Discuss different scopes of code optimizations and describe their applicability.
(b) Optimize the following code and discuss each optimization technique that you have applied stating their advantages.

```
#include <stdio.h>
int main() {
int j, n, array[10], k=1;
n=5;
n=k;
for(j=0; j< 3; j++) { array[n+j] = j*5; }
```

```
if (n>=5) array[n] = 5;
k = print();
return 0;
}

int print(){
printf("Hello World !\n");
return 0;
}
```

(c) What is an *Activation Record*? Provide a general structure of it. What is an *Activation Tree*?
(d) Briefly discuss the ***Mark and Sweep*** algorithm for garbage collection.

4+6+4+6=20

8. (a) What is a basic block? Write an algorithm to identify a basic block.
(b) Discuss with examples why you need to store the liveness and next-use properties of variables.
(c) Consider the following code:

1: $t_1 = a * a$
2: $t_2 = a * b$
3: $t_3 = 2 * t_2$
4: $t_4 = t_1 + t_3$
5: $t_5 = b * b$
6: $t_6 = t_4 + t_5$
7: $X = t_6$

Using the algorithm for deciding liveness and next-use information, discuss how do you decide the liveness and next-use of the variables and how do you decide about register reuse.
(d) Discuss when do you apply Loop Fusion and Loop Fission.

4+4+8+4=20