Ex/CSE/T/321/74/2012

3rd Year, Comp. Sc. & Engg. 2nd Semester Examination, 2012
Compiler Design

Time – 3 Hours **Full Marks – 100**

Use separate answer scripts for two parts
Answer all parts of a single question on contiguous pages

Part-I

Answer question 1 and any two from the rest

1.

a. Describe functioning of the lexical analysis phase of a compiler.
b. State necessity of "input buffering" in implementation of lexical analyzers.
c. What is an "activation record"?
d. What do you mean by the terms *l-value* and *r-value*. Give suitable examples.
e. What is "dangling reference"? Give an example.

2+2+2+2+2=10

2.

a. Given that binary number strings are read with the most significant bit first and may have leading zeroes, do the following for the language consisting of binary number strings that are powers of 2.
   i. Derive the regular expression describing the above language.
   ii. Use Thompson's construction to convert the above regular expression into an NFA.
   iii. Convert the NFA to equivalent DFA using subset construction.
b. What is a regular definition? Derive the regular definition for the identifiers in C programming language.
c. Describe functioning of code optimization phase of a compiler. Give suitable example.

(2+3+7)+(2+3)+3=20

3.

a. Describe the activities performed in the calling sequence and return sequence when function main() calls function a().

```
void a(int x, int y, int z){
        ...
}

int g(int x, int y){
        ...
}

main(){
        int i, j;...
        a(i, j, g(i+1, j+1));
        ...
}
```

b. Define static scope and dynamic scope. Determine output of the following program snippet considering both static and dynamic scope. Give justifications for your answer.

```
int x = 8;

void printx(void) {
        printf("%d\n", x);
}

void foo(int y) {
        int x = 6;
        x = x + x * y;
        printx();
}

void main() {
        int z = 4;
        printx();
        foo(z);
}
```

c. Explain how static or lexical scope is implemented in case of programming languages which supports nested procedure definition.

d. What is a display? Explain one method for its implementation.

7+5+4+4=20

4.

a. Show the activation record for a call to function f. Determine the offset of the identifiers x, c, f, a[4][1], and y with respect to frame pointer (fp). Assume that four, four, one, and eight bytes are required for storage of integers, addresses, characters and double-precision floating point numbers and row major storage for multidimensional array.

```
int f(char c[][3], double f, int x[]){
        int a[5][2]; double y;
        .....
}
```

b. Determine output of the following program using each of the parameter passing methods call by value, call by reference, call by name, call by value result. Give justifications for your answer.

```
int i=4;
int a[10]={0, 1, 2, 3, 4, 4, 3, 2, 1, 0};
void swap(int x, int y){
        x=x+y;   y=x-y;   x=x-y;
}
```

```
main(){
        swap(i, a[i]); printf("%d\n", i);
        for(int j=0; j<10; j++)
                printf("%d ", a[j]);
}
```

c. Explain the "Deep Access" and "Shallow Access" method for implementing dynamic scope. Give examples of each.

6+10+6=20

Part-II

Answer question no. 1 and any two from the rest

1. Answer any five questions. 5x2=10
   a. Define context-sensitive grammars.
   b. Arrange the following parsing techniques according to their processing power: SLR, Canonical LR and LALR.
   c. Give an example of static single assignment form.
   d. What types of attributes are supported in L-attributed definitions? Explain your answer.
   e. Give an example of Loop Fission. Explain how it improves the code.
   f. Briefly discuss how a backtracking parser works.

2. Consider the grammar
   E → E + T | T
   T → T F | F
   F → F* | *a* | *b*
   a. Find the FIRST and FOLLOW sets of the above grammar.
   b. Construct the LR(0) item sets and then SLR parsing table for the above grammar.
   c. Write at least one string (with not less than five tokens) that can be generated from the above grammar along with the leftmost and rightmost derivations of the string.
   d. Explain the differences between a parse tree and an abstract syntax tree.

   4+10+4+2=20

3.
   a. Construct an SLR(1) parsing table for the following grammar.
   S → id | V := E
   V → id
   E → V | n
   What kind of conflict do you find? Explain why.
   b. Translate the arithmetic expression 2*a+b+2*a+b into:
   (i) Directed Acyclic Graph, (ii) Quadruple, (iii) Triple.
   c. Discuss the different scopes of code optimisation techniques.

   8+2+6+4=20

4.
   a. Consider the grammar:
   A → B C *x* | *y*
   B → *y* A | ε
   C → A *y* | *x*
   In the above grammar, {x, y} is the set of terminal symbols. Is the grammar LL(1)? Justify your answer.
   b. Following is a grammar for *dangling else*
   stmt → if | **other**
   if → if **(** exp **)** stmt | if **(** exp **)** stmt **else** stmt
   exp → **0** | **1**
   Show that the grammar is ambiguous. Rewrite the grammar to remove ambiguity.
   c. Construct a LL(1) parsing table for the following grammar:
   S → + S S | * S S | a
   Draw a parse tree for the string + * a a a

   5+5+8+2=20

5. Consider the following grammar:

   D → T L

   T → int | float

   L → id, L | id

   a. Left factor the grammar
   b. Construct the FIRST and FOLLOW sets for the non-terminals of the resulting grammar.
   c. Show that the grammar is LL(1).
   d. Construct the LL(1) parsing table for the resulting grammar.
   e. Show the actions of the corresponding LL(1) parser, given the string **int x, y**.
   f. Explain how attributes can be associated with this grammar for deriving the types of the identifiers.

2+4+2+4+4+4=20

--------|-------