BACHELOR OF COMPUTER SCIENCE ENGG. EXAMINATION, 2009

(Third Year, Second Semester)

COMPILER DESIGN

Time : Three hours

Full Marks : 100

Answer any five questions taking at least two questions from each part
Use separate answer scripts for two parts
Answer to all parts of a single question must be written in contiguous pages

Part-I

1.

a. Describe functioning of the lexical analysis phase of a compiler.
b. State necessity of "input buffering" in implementation of lexical analyzers.
c. Derive the regular expression for a regular language consisting of set of strings that has exactly 2 number of b's and any number of a's. The alphabet is {a, b}.
d. Construct an equivalent NFA for the above regular expression, using Thompson's Construction.
e. Convert the above NFA to a DFA using Subset Construction algorithm

4+3+2+3+8=20

2.

a. Describe functioning of code optimization phase of a compiler. Give suitable example.
b. What is a regular definition? Derive the regular definition for the identifiers in C programming language.
c. Describe how C compiler handles variable number of arguments to functions like printf, scanf etc.
d. 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 = 4;

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

void foo(int y) {
        int x = 4;
        x = x + x * y;
        printx();
}

void main() {
        int z = 3;
        printx();
        foo(z);
}
```

e. Describe with an example "most closely nested" rule in determining scope of a declaration in a block-structured language.

3+3+4+6+4=20

3.

a. Describe briefly the stack and heap storage allocation strategy.
b. Describe how variable length data items are handled in stack storage allocation strategy.
c. State the reasons of having an intermediate representation of the source program during compilation.
d. What do you mean by the terms *l-value* and *r-value*. Give suitable examples.
e. Give output of the following program using each of the four parameter passing methods call by value, call by reference, call by name, and call by value result. Give justifications for your answer.

```
int i=0;

void swap(int x, int y)
{
        x=x+y;
        y=x-y;
        x=x-y;
}

main()
{
        int a[3]={1, 2, 0};
        swap(i, a[i]);
        printf("%d %d %d %d\n", i, a[0], a[1], a[2]);
        return(0);
}
```

4+3+2+3+8=20

4.

a. What is "dangling reference"? Give an example.
b. What do you mean by activation of a procedure/function? Draw the activation tree of the following program snippet.

```
int fib(int num){
        switch(num) {
                case 0:
                        return(0);
                        break;
                case 1:
                        return(1);
                        break;
                default:
                        return(fib(num - 1) + fib(num - 2));
                        break;
```

```
        }
}

main(){
        fib(5);
}
```

c. What is an "activation record"? Describe different fields of an "activation record".

d. Consider the following C function.

```
int f(int x[], char c)
{
        int a[12];
        double y;
        ……..
}
```

Show the activation record for a call to f. Determine the offset of the variables x, c, a[8], 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.

4+4+4+8=20

Part-II

1. Consider the following grammar:

   $E \rightarrow (L) \mid a$

   $L \rightarrow E , L \mid E$

   [redacted] LALR(1) items for this grammar.

   b. Construct the SLR(1) parsing table.

   c. Show the parsing stack and the actions of an SLR(1) parser for the input string ((a) a (a a))

   d. Construct the DFA of LALR(1) items for the above grammar.

   5+5+5+5 = 20

2. Consider the grammar

   stmt_seq → *stmt* ; *stmt_seq* | *stmt*

   stmt → **s**

   a. Left factor the grammar. Construct the FIRST and FOLLOW sets for the left-factored grammar. Construct the LL(1) parsing table for the above grammar.

   b. Give a leftmost derivation for the string "**s ; s ; s**".

   c. What are attribute grammars? What are the different types of attributed used in attribute grammars? Explain S-attributed and L-attributed definitions with examples.

   (2+4+4)+2+(2+2+4) =20

3.

   a. Consider the grammar $S \rightarrow S\,S + \mid S\,S\,* \mid a$. Give a rightmost derivation for the string " a a + a * " and draw the parse tree. Is the grammar ambiguous? Justify your answer.

   b. Consider the grammar $S \rightarrow S\,(\,S\,)\,S \mid \varepsilon$. Give a leftmost derivation for the string " (( ) ( ))" and draw the parse tree. Construct the predictive parsing table for the grammar (after left-factoring or removing left recursion if necessary).

c. Discuss error recovery in predictive parsing.

(3+2)+(3+8)+4=20

4.

a. What are the purposes of using a symbol table?

b. Write the syntax-directed definition (semantic rules) for the floating point value of a decimal number given by the following grammar:

dnum → *num.num*

num → *num digit* | *digit*

digit → **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9**

(Use a *count* attribute to count the number of digits to the right of the decimal point.)

c. Draw an annotated parse tree and a dependency graph for parsing the integer 48.23.

d. Construct FIRST and FOLLOW sets for the above grammar.

e. Find LR(0) sets of items for the above grammar.

2+4+4+4+6=20

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