Department of Computer Science and Engineering
B.Sc. Engg, Part-III, Odd Semester, Examination-2021
Course: CSE3141 (Compiler Design)
Full Marks-52.5 Time: 3 hours

[N.B. Answer any SIX questions taking THREE from each of the sections]

Section-A

1. a) What is the difference between a compiler and interpreter? 2
   b) Suppose a source program contains the assignment statement, 5
   position = initial + rate * 60
   Explain how this statement is processed and finally translated at different phases of a traditional compiler.
   c) Distinguish between single-pass and multi-pass compiler. 1.75

2. a) Define token and lexeme. What are the functions of lexical analyzer? 3
   b) Explain the necessity of regular expression and context-free grammar in designing of a compiler. 2
   c) Construct a lexical analyzer (*i.e.* DFA) for the regular expression: (a|b)*b(a|b). 3.75

3. a) Write down formal definition of grammar. Discuss Chomsky hierarchy of formal grammars. 3.75
   b) Define normal Chomsky Form (CNF). Convert following CFG into Chomsky Normal Form : 5
   i) S→ ASaBA ii) A→B|S iii) B→b|ε

4. a) Define ambiguity of context-free grammar. Using disambiguation rule make the grammar unambiguous: 2.5
   E→(E)| E-E|E*E|E+E|id
   b) Define LL (1) grammar. Convert the above grammar into LL (1). Construct a predictive parsing table using the grammar. 6.25

Section-B

5. a) What is compiler? Define the types of compilers. 3
   b) Briefly discuss the functional components of a compiler. 3.5
   c) What is front end and back end of a compiler? Why the compilers' functional components (phases) should be divided? 2.25

5. a) What is bottom-up parsing? How it is implemented? 1.5
   b) Construct an operator relation table for operator precedence parser for the following grammar: 2.25
   E→ EAE|-E|id, A→+|*.
   c) Check following grammar SLR(1) or not: S→T, T→T*F|F, F→id. 5

7. a) Explain syntax-directed translation (SDT) scheme. 2
   b) Write down SDT for following CFG: 4.5
   S → id = E, E → E-T | T, T → T/F | F, F → id
   using the required SDT produce three-address code for the statement "x=a-b/c".
   c) Write down a postfix notation for the infix statement "if *a* then if *c-d* then *a+c* else *a*c* else *a+b*". 2.25

8. a) Define code optimization. What are the principal sources of optimization? Explain in detail. 3.75
   b) What do you mean by local and global optimization? Shortly discuss these two phases of optimization. 5

University of Rajshahi

Department of Computer Science & Engineering

B.Sc. (Engg.) Part-3 Odd Semester Examination- 2020

Course Code: CSE 3141

Course Title: Compiler Design

Full Marks: 52.5 **Time: 3 Hours**

[Answer any SIX (06) questions taking THREE (03) from each section]

Section A

1. a) Figure 1(a) shows four different ways to translate source code written in a high level language into machine code. Write which kind of translator can be used in each case. What are the positive sides of each translator? 5.00

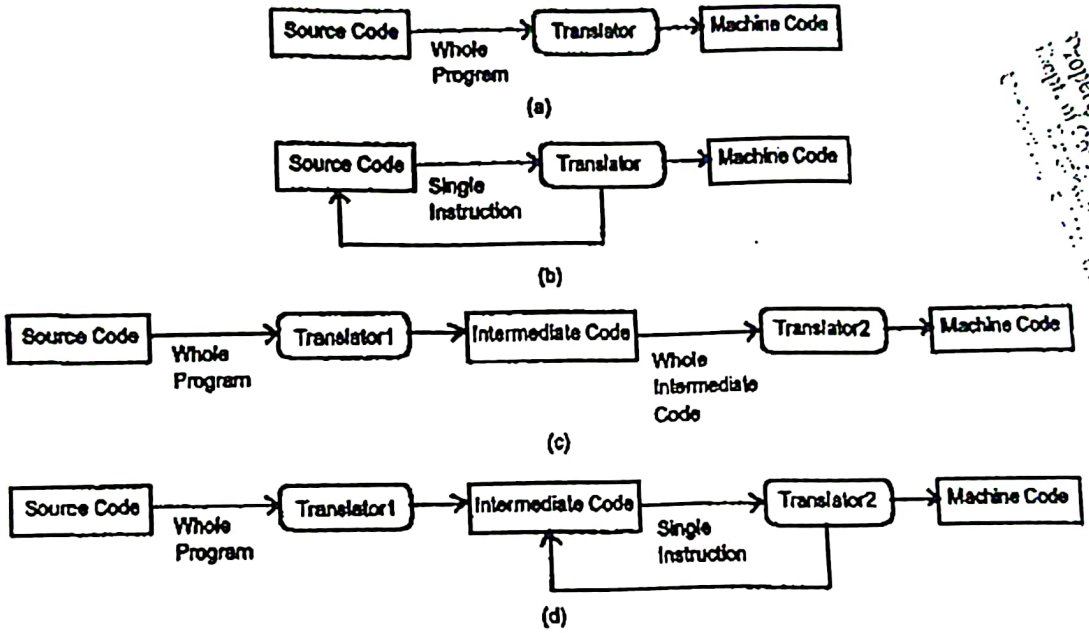


Fig. 1(a)

b) How can be a compiler written in the source language that it intends to compile? 3.75

2. a) Define token, lexeme, and pattern with examples. 3

b) State the role of lexical analyzer with a neat diagram. Identify the lexemes and their corresponding tokens in the following statement: 3.75

printf ("Simple Interest=%f\n", si);

c) Consider the context free grammar 2

$S \rightarrow aSbS|bSaS|\varepsilon$

Check whether the grammar is ambiguous or not; why- Explain.

3. a) How can you eliminate left-recursion from a grammar? Eliminate left recursion from the following grammar: i) $S \rightarrow Aa|b$ ii) $S \rightarrow Ac|Sd|\varepsilon$ 2.75

b) What do you mean by FIRST and FOLLOW sets? Compute the FIRST (F), FIRST (T'), FIRST (T), FIRST (E'), and FIRST (E) from the following grammar given below: 6

$E \rightarrow TE'$

$E' \rightarrow +TE'|\varepsilon$

$T \rightarrow FT'$

$T' \rightarrow *FT'|\varepsilon$

$F \rightarrow (E)|$ id

4. a) Sometimes left factoring is needed; why? The following grammar abstracts the "dangling-else" problem: S→iEtS|iEtSeS|a 3
E→b
Here, *i*, *t*, and *e* stand for **if**, **then**, and **else**; *E* and *S* stand for **"conditional expression"** and **"statement."** What will be left-factored of this grammar?

b) Write an algorithm for predictive parsering table. 3

c) What are the rules of type checking? Briefly illustrate how can type conversions happen? 2.75

Section B

5. a) A Syntax-Directed Translation scheme that takes strings of a, b, and c as input and produces as output the number of substrings in the input string that correspond to the pattern a(a|b)*c+(a|b)*b. For example, the translation of the input string "abbcabcababc" is "3". 4
(i) Write context-free grammar that generates all strings of a, b, and c.
(ii) Give the semantic attributes for the grammar symbols.
(iii) For each production of the grammar, present a set of rules for evaluating the semantic attributes.

b) Differentiate between S-attributed SDT and L-attributed SDT with suitable examples. 2.75

c) Explain common sub expression elimination with an example. 2

6. a) Why is a symbol table needed? Write down the purpose and operations of the symbol table. 4

b) Write the basic blocks and draw control flow graph of the source code: 2.5

```
w =0;
y =0;
if (x > z)
 {
    y = x;
    x++;
  }
else
  {
    y=z;
    z++;
  }
w= x+z;
```

c) What do you mean by dead code elimination? 2.25

7. a) Write down differences between the parse tree and syntax tree with proper examples. 2

b) Draw the syntax tree and parse tree of the expression: 3
(A+B/C)/(A-C/F)*F+(H*Y*Z)

c) Translate the arithmetic expression A:= B+(C*D) into: 3.75

i) Quadruples
ii) Triples
iii) Indirect Triples

8. Consider the following fragment of code, it computes the dot product of two vectors **x** and **y** of length 10:

```
begin
    a := 0
    b := 1
    do
        begin
            a := a + x[b] + y[b]
            b := b + 1
        end
    while b <= 10
end
```

a) Write three-address code for the above fragment code for a machine with four bytes/word. 2.75

b) Draw the flow graph of the three-address code having two induction variables. 3.00

c) Draw the flow graph of the three-address code after eliminating one induction variable. 3.00

University of Rajshahi
Department of Computer Science & Engineering
B.Sc.(Engg.) Part-3 Odd Semester Examination- 2019
Course Code: CSE 3141 Course Title: Compiler Design

Full Marks: 52.5 **Time: 3 Hours**

[N.B. Answer any SIX questions taking THREE from each section.]

SECTION- A

1. a) Define compiler. 1
   b) What are the cousins of the compiler? Explain them. 2.75
   c) Suppose a source program contains the assignment statement, 5

   $$a = initial - rate + rate * 60$$

   Explain how this statement is processed and finally translated at different phases of a traditional compiler with neat diagram.

2. a) Explain parse tree. Write down the properties of the parse tree. 2.75
   b) Define ambiguity. How can you remove ambiguity from arithmetic expression? 2
   c) Consider the context-free grammar: 4

   (i) $E \rightarrow I \mid E+E \mid E-E \mid E*E \mid (E)$ (ii) $I \rightarrow a \mid Ia$

   And the string is (a-a)*a+a
   a) Give a leftmost and rightmost derivation for the string.
   b) Give a parse tree for the string.
   c) Is the grammar ambiguous or unambiguous? Explain.

3. a) Define finite automata (FA). How does a DFA Contribute to design a compiler? 2.75
   b) Construct DFAs accepting the following languages: 6
   (i) $\{w \in \{a,b\}^* \mid w$ has at most two a's$\}$
   (ii) $\{a^m b^n \mid m, n \geq 1\}$
   (iii) $\{w \in \{0, 1\}^* \mid w$ is a binary number divisible by 3$\}$

4. a) Define LL(1) grammars. 1.75
   b) Consider the following grammar: 7

   $$S \rightarrow iEtS \mid iEtSeS \mid a$$
   $$E \rightarrow b$$

   (i) Compute FIRST(S) and FOLLOW(S)
   (ii) Construct the predictive parsing table

SECTION- B

5. a) "The top-down parsing method cannot handle left-recursive grammar" explain with suitable example. 2

   b) Consider the following grammar: 4

$$E \rightarrow E+T \mid T$$
$$T \rightarrow T*F \mid F$$
$$F \rightarrow (E) \mid id$$

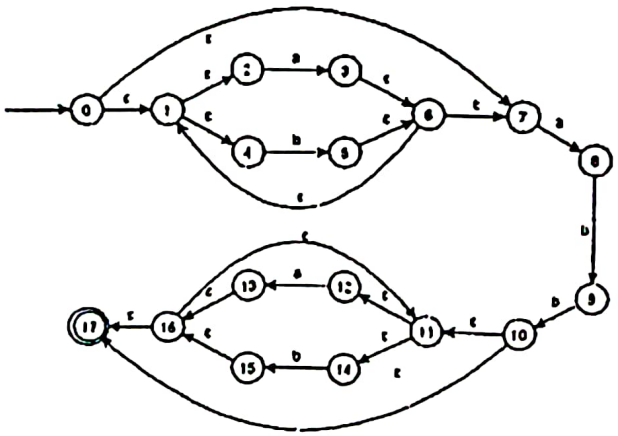
   And the string is id+id*id

   Give bottom-up parses for the input string by using shift-reduce parser.

   c) Define alphabet, string and language. 2.75

6. a) Define symbol table. 1.75

   b) Convert the following NFA to DFA. 7



7. a) Define syntax-directed translation (SDT) scheme. What is three-address code? 2

   b) Write down SDT for the following CFG: 4.75

   S→id=E, E→ $E + T \mid T$, T→T*F | F, F→id

   Using the required SDT produce three-address code for the statement "x=a+b*c".

   c) Explain syntax tree with example. 2

8. a) What is transition diagram? Draw a transition diagram for whitespace. 1.75

   b) Discuss about the role of lexical analyzer. 3

   c) Write the names of error recovery strategies and explain them. 4

University of Rajshahi

Department of Computer Science and Engineering

B.Sc. (Engg.) Part III Odd Semester Examination-2018

Course: CSE3141 (Compiler Design)

Full Marks-52.5 Time: 3 hours

[N.B. Answer any SIX questions taking THREE from each section]

Section-A

a) Define compiler. Distinguish between single-pass and multi-pass compiler. Write down the differences between compiler and interpreter. 4

b) What are the advantages to group the phases of compiler into front end and back end? 2

c) Describe the different categories of errors encountered by the different phases of compiler. 2.75

2. a) What do you mean by sentence of a grammar? Justify whether the string –(id+id) is a sentence of the grammar E → E+E | E*E | (E) | -E | id or not. 2.75

b) Define parse tree. Describe the construction of parse tree for a sentence 'cad' considering the grammar: (i) S → cAd (ii) A → ab|a. 3

c) Explain how can you eliminate left-recursion from a grammar. Eliminate left-recursion from: (i) S→Aa|b (ii) A→Ac|Sd|ϵ. 3

3. a) Construct DFAs accepting the following languages: 6

i) {w ϵ {a, b}*| w starts and ends with same symbol}

ii) {w ϵ {a, b}*| w has even number of a's and b's}

iii) { w ϵ{0,1}* | w is a string interpreted as a binary number ≡ 1 mod 4}

b) Why NFA is not directly implemented in designing lexical analyzer compared to DFA? 2.75

4. a) What is NFA? Write down the formal definition of ε-NFA. Why ε-NFA is necessary to design a lexical analyzer? 3.75

b) Construct an FA for the regular expression ab(b|c)*abb applying Thomson's construction. How many number of states are necessary to construct a finite automata for that regular expression? 5

Section-B

5. a) Define context-free grammar. Classify context-free grammar in various aspects. 2

b) What do you mean by ambiguous grammar? Prove that the following grammar is ambiguous: S→iEtS|iEtSeS|a, E→b. 3

c) Discuss the difficulties of top-down parsing? Eliminate non-determinism from the following grammar: S→ aSSbS | aSaSb | abb | b 3.75

6. a) Define LL(1) grammar. Why LL(1) parser is known as predictive parser? 1.75

b) Construct a parsing table for predictive parser with the following CFG grammar by first eliminating left recursion (if necessary): 7

S → A, A → aB | Ad | B → bBC | f, C → g

7. a) Difine syntax-directed translation (SDT) scheme. What is three-address code? 2

b) Write down SDT for following CFG: 4.75

S → id = E, E → E+T | T, T → T*F | F, F → id

using the required SDT produce three-address code for the statement "x=a+b*c".

c) Evaluate postfix notation for the statement "if *a* then if *c-d* then *a+c* else *a*c* else *a+b*" 2

8. a) Write the importance of symbol table. Also mention various operations on symbol table. 3

b) Draw the syntax tree for the expression: (A+B/C)/(A-C/D)*D+(X*Y*Z). 2

c) Write the quadruple and triple representation for the expression: A:=-B*(C+D). 3.75

University of Rajshahi

Department of Computer Science and Engineering

B.Sc. (Engg.) Examination-2017, Year-III, Semester-I

Course: CSE3141 (Compiler Design)

Full Marks-52.5 Time: 3 hours

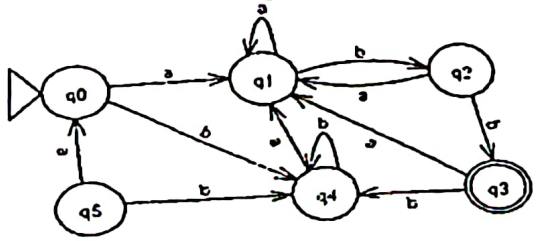
[N.B. Answer any six questions taking THREE from each of the groups]

Part-A

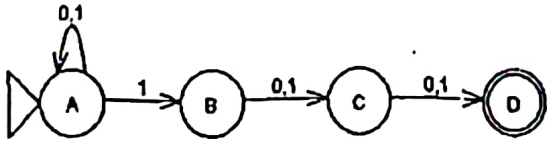
1. a) What are the phases of the compiler? Explain with neat diagram. 4.75
   b) Discuss the functions of lexical analyzer. 2
   c) Differentiate between single-pass and multi-pass compiler. 2

2. a) Describe the constructing rules for regular expressions. 2.75
   b) Convert the following regular grammars to regular expressions: 4
      i) S→aS, S→aB, B→bC, C→aC, C→ a.
      ii) S→aA, S→ a, A→aA, A→bB, A→a, B→bB, B→aA.
   c) Discuss the importance of regular expression and context-free grammar in designing a compiler. 2

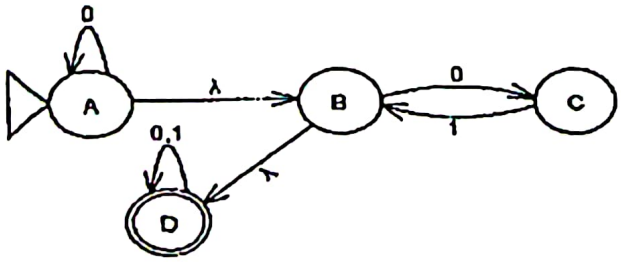
3. a) Define DFA. How does a DFA contribute to design a compiler? 1.75
   b) Construct DFAs accepting the following languages: 4
      i) {w ∈ {a, b}*| w starts and ends with different symbol}
      ii) {w ∈ {a, b}*| w has odd number of a's and b's}
   c) Minimize the following DFA: 3



4. a) Build a DFA that simulates the following NFA: 4.75



   b) Define ε-NFA. Convert the following ε-NFA to NFA (where, lambda stands for ∈): 4



Part-B

5. a) What do you mean by ambiguous grammar? Disambiguate the following CFG: 5
   i) A → A@B | B, B → B#C | C, C → C@D | D, D →d
   ii) E → E-E | E*E | E^E|-E
   b) What is left-factoring? Why it is necessary in top-down parsing? 1.75
   c) Eliminate non-determinism from the following grammar: 2
   S→aSSbS | aSaSb | abb | b

6. Given the grammer: (i) E→E+T|T (ii) T→T* id|id.
   a) Construct sets of LR(1) items 5
   b) Construct canonical LR(1) parsing table 2.75
   c) Is the grammar LALR(1) ? Justify your answer. 1

7. a) Distinguish between parse tree and syntax tree with example. 2
   b) Draw the syntax tree of the expression: (A+B/C)/(A-C/F)*F+(H*Y*Z). 2.75
   c) Give the quadruple and triple representation for the statement: A:=-B*(C+D). 4

8. Write down short note (any three): 8.75
   a) Symbol table
   b) Loop optimization
   c) Quadruple
   d) Operator grammar
   e) LALR(1) parser

Department of Computer Science and Engineering
B.Sc. Engg. Part-III Odd Semester Examination-2016
Course: CSE3141 (Compiler Design).

Time: 3 Hrs. Full Marks: 52.5

[Answer any **six (06)** questions taking **three (03)** from each section.]

Part A

1. (a) What is the benefit of generating intermediate code before generating the object code? 1.75
   (b) For the following **while-statement**, draw a parse tree and write down the intermediate code. 7

$$while((a > c)\ \&\&\ (a <= 2 * (b - 15)))$$
$$a = a * (b - 15);$$

2. (a) Write down the symbol table contents for the following C program after parsing phase: 2.75

```
int a, b;
int add(){
   int c;
   c = a + b;
   return c;
}
main(){
   int c;
   a = 10; b = 20;
   if (a > 10)
    c = add();
}
```

   (b) There are many techniques for building a symbol table, among which hashing is quite popular. In this technique, a hash function is used for generating addresses of identifiers. Write down the disadvantages of the following hash functions: 6

(i) $h(identifier) = x_1$

(ii) $h(identifier) = x_1 a^{n-1} + x_2 a^{n-2} + \ldots\ldots + x_n a^0$

(iii) $h(identifier) = x_1 \ mod \ N$

(iv) $h(identifier) = (x_1 a^{n-1} + x_2 a^{n-2} + \ldots\ldots + x_n a^0) \ mod \ N$

Assume that:
x_i is the ASCII value of the i-th left-most character of the identifier.
$a = \{33, 37, 39, 41\}$ and N is a prime number.

3. (a) Using the following grammar, show the steps of shift-reducing parsing for the following C instructions. 5.75

Grammar:

S --> S';
S' --> *identifier* = E | ε
E --> E + T | E - T | T
T --> T * F | T / F | F
F --> -E | (E) | *identifier* | *number*

Instructions:

```
x = -y + 123 / (5 + 3 * z);
x = y + 123 */ (3 + z);
```

(b) Write down the *context free grammar* (CFG) for the following stream of *printf()* statements and *if-statements*: 3

i) *printf ("%d",a); | printf (""); | printf ("Compiler"); |*
printf ("%f",a); | printf ("%f%d",a,b); | printf (%f,%d,a,b); |
printf ("%f %d",a,b);

ii) *if (a>0) printf (%d,a); | if (a>0) {printf (%d,a); |*
if (a>0); | if (a>0) {} | if (a>0) {;}

4. (a) Define LL(1) grammar. 1.75

(b) Define a context-free grammar which describes the same language as the regular expression (a / b) * ab. 3

(c) What is left-recursion in a grammar? "The top-down parsing method cannot handle left-recursive grammar" – explain. 4

Part B

5. Consider the following grammar:

Grammar:

S --> S';
S' --> I = E | ε
E --> E + T | E - T | T
T --> T * F | T / F | F
F --> -E |(E) | I | C
I --> *identifier*
C --> *number* | *string* | *character*

(i) Compute FIRST(A) and FOLLOW(A). 3.5

(ii) Construct the predictive parsing table. 3.5

(iii) Show whether the grammar is LL(1). 1.75

6.(a) Distinguish between parse tree and syntax tree with example. 2

(b) Draw the syntax tree for the expression: (A + B / C) / (A – C / F) * F + (H * Y * Z). 2.75

(c) Give the quadruple and triple representation for the statement: A := – B * (C + D). 4

7.(a) What do you understand by *Ambiguous Grammar*? 1.75

(b) Which of the following sentences can be derived from the given grammar with starting nonterminal S? In each case, give a 7

(i) leftmost derivation
(ii) rightmost derivation
(iii) derivation tree

Grammar:

S --> aAcB | BdS
B --> aAcA | cAB |b
A --> aB | aBc | a

Sentences:

(a) *aacb* (b) *abcaababcd*

8.(a) Why do we need symbol table? Illustrate various operations on symbol table. 4

(b) Briefly discuss code optimization procedure. 3

(c) Explain the necessity of intermediate code generation. 1.75

Rajshahi University

Department of Computer Science and Engineering

B.Sc. (Engg.) Part-III, Odd Semester Examination-2015

Course: CSE-3141 (Compiler Design)

Full Marks: 52.5, Time: 3 hours

Answer any Six (06) questions taking Three (03) from each part:

PART A

1. a) What is the difference between a compiler and interpreter? 2

   b) Suppose a source program contains the assignment statement, 5

   position = initial + rate * 60

   Explain how this statement is processed and finally translated at different phases of a traditional compiler.

   c) Distinguish between single-pass and multi-pass compiler 1.75

2. a) Define token, lexeme and pattern. Give examples. 3

   b) Let L be the set of letters {A, B, . . . , Z, a, b, . . . , z} and let D be the set of digits {0,1,. . .9}. We may think of L and D in two, essentially equivalent, ways. One way is that L and D are, respectively, the alphabets of uppercase and lowercase letters and of digits. The second way is that L and D are languages, all of whose strings happen to be of length one. Then describe the languages for $L \cup D$, LD, L^4, L^*, $L(LUD)^*$ and D^5. 3

   c) What do you mean by 'alphabet' and 'string'? Discuss the various string operations. 2.75

3. a) Differentiate DFA and NFA. Give examples. 3

   b) For the given regular expression (a|b)*a, draw its NFA and then convert NFA to the equivalent DFA using subset construction method. 5.75

4. a) What are the rules for type checking? How are the type conversions happened? Explain. 3.75

   b) Write down the unifications algorithm. Hence, simulate the algorithm for at least one example. 5

PART B

5. a) Define parse tree. Describe the construction of parse tree for a sentence 'cad' considering the grammar: (i) S→ cAd (ii) A→ab|a. 3.75

b) Define Left-recursive grammar. How can you eliminate Left-recursion from a context-free grammar? Elimminate Left-recursion from: i) S→Aa|b ii) A→ Ac|Sd|ε. 5

6. a) What is the difference between left-most and right-most derivations? Give examples. Drive the string (id+id)/(id-id) for both the derivation methods following the grammar rules: 2.75
E→ E+E| E-E| E*E| E/E|(E)| id

b) How do you differentiate a sentential form with a sentence of a grammar? Explain with examples. 2

c) Write down the construction rules for constructing a grammar from a regular expression. Hence, show that the regular expression (a|b)*abb and the following grammar describe the same language. 4

$A_0 \rightarrow aA_0 | bA_0 | aA_1$

$A_1 \rightarrow bA_2$

$A_2 \rightarrow bA_3$

$A_3 \rightarrow \varepsilon$

7. a) What are the differences between Top-Down and Bottom-Up parsers? Explain with examples. 2.75

b) What do you mean by FIRST and FOLLOW sets? Write an algorithm to compute the FIRST of $(X_1X_2X_3.....X_N)$. Hence, compute the FIRST (F), FIRST (T'), FIRST (T), FIRST (E') and FIRST (E) following the grammar given below: 6

$E \rightarrow T\ E'$

$E' \rightarrow +T\ E' \mid \varepsilon$

$T \rightarrow F\ T'$

$T' \rightarrow *F\ T' \mid \varepsilon$

$F \rightarrow (\ E\ ) \mid id$

8. a) Define code optimization. What are the principal sources of optimization? Explain in detail. 3.75

b) What do you mean by local and global optimization? Shortly discuss these two phases of optimization. 5

University of Rajshahi
Department of Computer Science and Engineering
B.Sc. (Engg.) Part-III Odd Semester Examination-2014
Course: CSE 3141 (Automata Theory and Compiler Design)
Full Marks: 52.5 Time: 3 Hours

(Answer any six questions not taking more than three from each group)

Group A

1.(a) Define compiler. What are the differences between compiler and interpreter? 2.75

(b) Discuss the process of token generation using input buffering system. 3

(c) Explain the role of Syntax analyzer with example. 3

2.(a) Differentiate token, pattern and lexeme. Give examples. 3

(b) What are the common classes of tokens used in most of the programming languages? Explain. 1.5

(c) Explain with an example how the language operations L^* and L^+ differ? 1.5

(d) If r and s are two regular expressions denoting the languages $L(r)$ and $L(s)$, then what do the language operations $(r)|(s)$, $(r)(s)$, $(r)^*$ and (r) mean? 2

(e) What does the regular expression (a|b)(a|b) denote? 0.75

3.(a) What is transition diagram? Draw a transition diagram for recognizing the relational operators described in C programming language. 2.5

(b) Define transition table for NFA. Draw the NFA which corresponds the adjacent transition table. 2.25

STATE	a	b	ϵ
0	{0,1}	{0}	∅
1	∅	{2}	∅
2	∅	{3}	∅
3	∅	∅	∅

(c) Describe the McNaughton-Yamada-Thompson algorithm to convert a regular expression to an NFA. Hence construct the corresponding NFA for the regular expression $a|b^*c$. 4

4.(a) Describe the role of a parser in a traditional compiler. 3

(b) Give the formal definition of context-free grammar. 2.25

(c) Consider the grammar S→iE+SŚ|a, Ś→e S|ε, E→b and prove that, this grammar is ambiguous for non-recursive predictive parsing. 3.5

Group B

5.(a) What do you mean by a sentence of a grammar? Why do you say the string –(id+id) is a sentence of the grammar E→E+E|E*E|(E)|-E|id ? Explain. 2.25

(b) What is the difference between a sentence and sentential-form? Explain with examples for both left-most and right-most derivations. 2

(c) What are the rules for constructing a grammar from NFA? Define a context-free grammar which describes the same language as the regular expression (a|b)*ab. 4.5

6.(a) What is left recursion in a grammar? "The top-down parsing method cannot handle left-recursive grammar"- explain with suitable example. 3

(b) What are the advantages of left-factoring a grammar? Draw an algorithm for left-factoring a grammar and hence left-factor the abstract form of the grammar representing 'dangling-else' problem. 5.75

$$S \rightarrow i\ E\ t\ S \mid i\ E\ t\ S\ e\ S \mid a$$

$$E \rightarrow b$$

Where i, t and e stand for if, then and else; E and S stand for 'conditional expression' and 'statement'.

Ś→S, S→L=R|R, L→*R|id, R→L

7.(a) Define LL(1) grammar. 1

(b) If the following pre-computed parse-table is given for predictive parsing from the grammar E→TE', E'→+ TE' | ε, T→FT', T'→*FT' | ε, F→(E) | id; then parse the input string (id*id)+id. Hence show the stack contents, look-ahead symbols and sequence of the output rules and the operations applied. 6

	id	+	*	(	)	$
E	E→TE'			E→TE'		
E'		E'→+TE'			E'→ ε	E'→ ε
T	T→FT'			T→FT'		
T'		T'→ ε	T'→*FT'		T'→ ε	T'→ ε
F	F→id			F→(E)		

(c) Show the left-most derivations of the input string and construct the parse-tree from the above output rules. 1.75

3.(a) Briefly discuss how code optimization improves the execution efficiency of a program. 3.75

(b) What do you mean by local and global optimization? Explain the necessity of intermediate code generation. 5