University of Rajshahi

Department of Computer Science and Engineering
B.Sc. Engg, Part-III, Odd Semester, Examination-2023
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 a finite state machine? Construct a finite state machine that accepts all strings over the alphabet {a, b} that ends with the substring "ab". 3
   b) Explain the structure of a compiler with a neat diagram. Highlight the role of each phase in the compilation process 5.75

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 for Boolean expression unambiguous: E→ E∧E|E∨E|¬E|t|f 3
   b) Construct a LL(1) parsing table using the grammar: S→aABb, A→c|ε, B→d|ε, write down the sequence of moves by the LL(1) parser using the parsing table for the string 'adb'. 5.75

Section-B

5. a) What is an object program? How is it different from source code? 2.75
   b) List two major challenges in code generation 2
   c) Explain the machine model used in code generation. How does it influence the design of a code generator? 4

6. a) Define bottom-up parsing. Why bottom-up parsers are also known as shift-reduce parsers? Explain. 2.5
   b) Construct an operator relation table for operator precedence parser for the following grammar: 2
   E→ EAE|-E|id, A→+|*.
   c) Check following grammar SLR(1) or not: S→T, T→T*F|F, F→id. 4.25

7. a) Define intermediate code. Discuss various type intermediate codes often used in compiler. 3
   b) Explain syntax-directed translation (SDT) scheme. Write down SDT for following CFG: 5.75
   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".

8. a) What is semantic checking of a compiler? Write down some semantic errors of a computer program. 2
   b) A source code can be directly be translated into its target machine code. Then, why we need to translate the source code into an intermediate code first? Explain. 2.5
   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
   d) What is symbol table? Why it is essential in compiler design? 2.25

University of Rajshahi

Department of Computer Science and Engineering

B.Sc. (Engg.) Part-3 Odd Semester Examination-2022

Course: CSE3141 (Compiler Design)

Full Marks-52.5 Time: 3:00 hours

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

Section-A

1. a) Write down formal definition of grammar. Discuss Chomsky hierarchy of formal grammars. 2.75
   b) Define normal Chomsky Form (CNF). Convert following CFG into Chomsky Normal Form : 6
   i) S→ ASaBA ii) A→B|S iii) B→b|ε

2. a) What is parser? Classify parsers. 3
   b) Explain operator precedence parsing for an expression *a*+*b***c* with the following grammar: 2
   E→EAE|*a*|*b*|*c*, A→ +|*
   c) Construct operator function table for the above grammar. 3.75

3. a) Why bottom-up parser is also known as shift-reduce parser? 2.25
   b) Explain top-down and bottom-up parsing style with the following CFG with a sentence, '*abbcde*': 4
   S→aABe, A→Abc|b, B→d.
   c) Determine FIRST and FOLLOW sets for the following grammar: 2.5
   S→ABCDE, A→a|ε, B→b|ε, C→c, D→d|ε, E→e|ε.

4. a) Define NFA. Explain the formal definition of ε-NFA. Discuss the necessity of ε-NFA to design a lexical analyzer. 3
   b) Discuss finite automaton (FA). Construct an FA for the regular expression ab(b|c)*abb applying Thomson's construction. 3.5
   c) Define unambiguity. Explain null and unit production in theory of automata. 2.25

Section-B

5. a) Define compiler? Compare between compiler and interpreter. 2.25
   b) Explain the first three phases of a compiler with following source code: 4.5
   *x= -a+b*c; \\\\Arithmatic statement.*
   printf ("the value of x:", x);
   c) Why the phases should be divided? 2

6. d) What is translator? Classify translators based on level of programming languages. 2.75
   e) What is bootstrapping in compiler design? How it helps in self-compilation of a compiler? 3
   f) Given that there is a Pascal translator written in C language which translates any Pascal program to a corresponding C program. Using the concept of bootstrapping how to create that translator written in C++? 3

7. a) Explain context-free grammar. Classify context-free grammar in various aspects. 2
   b) Prove that the following grammar is ambiguous: S→iEtS | iEtSeS |a, E→b. 3
   c) Explain the elimination process of left-recursion from a grammar. Eliminate left-recursion from the following grammar: (i) S→Ba|b (ii) B→Bc|Sd|ε. 3.75

8. a) Why intermediate code is necessary for designing a compiler? 2
   b) Explain syntax-directed translation (SDT) scheme. Write down SDT for following CFG: 4.75
   E → E*T | T, T → T+F | F, F → id
   using the required SDT produce postfix notation for the expression, "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

University of Rajshahi

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

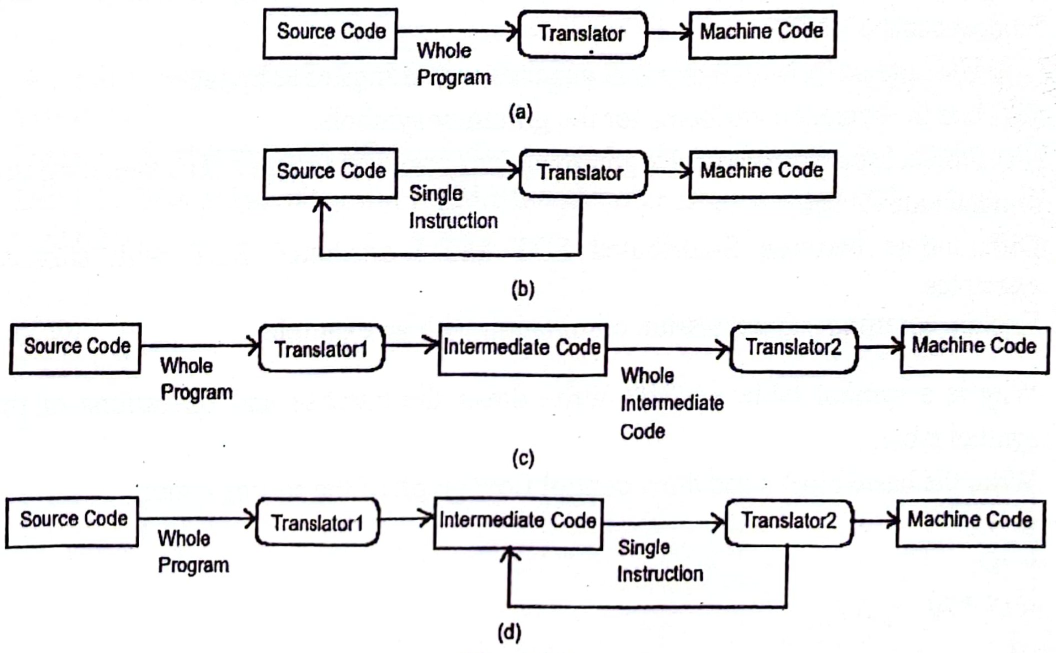

Fig. 1(a)

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

$S \to aSbS|bSaS|\varepsilon$

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) $S \to Aa|b$ ii) $S \to Ac|Sd|\varepsilon$ 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

$E \to TE'$
$E' \to +TE'|\ \varepsilon$
$T \to FT'$
$T' \to *FT'|\varepsilon$
$F \to (E)|\ id$

4. a) Sometimes left factoring is needed; why? The following grammar abstracts the "dangling-else" problem: $S \rightarrow iEtS|iEtSeS|a$ 3

$E \rightarrow b$

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