Department of Computer Science and Engineering
University of Chittagong
7^{TH} Semester BSc (Engg.) Examination 2023
CSE 711: COMPILERS
Marks: 54 Time: 4 hours

[Using section-wise separate answer scripts, answer any three questions from each of the following two sections and all the parts of a question chronologically.]

SECTION A

A-1/ (a) Write regular definitions for the following language: 2

i. unsigned numbers such as **3234**, **2334.34**, **454.34E5**, or **334.34E − 4**.
ii. all strings of lowercase letters that contain the five vowels in order

(b) What tasks are performed in the *front-end* and *back-end* of a compiler, and why? 3

(c) What is the role of symbol tables in compilers? Describe the components of a typical symbol table and outline its implementation using a hash table approach in **C**. 4

A-2/ (a) Consider the following grammar: 4

$$\begin{aligned} bexpr &\rightarrow bexpr \text{ or } bterm \mid bterm \\ bterm &\rightarrow bterm \text{ and } bfactor \mid bfactor \\ bfactor &\rightarrow \text{not } bfactor \mid (\ bexpr\ ) \mid \text{true} \mid \text{false} \end{aligned}$$

i. Left factor this grammar if necessary.
ii. Does left factoring make the grammar suitable for top-down parsing?
iii. In addition to left factoring, eliminate left recursion from the original grammar.
iv. Is the resulting grammar suitable for top-down parsing?

(b) How recursive-descent parsing works? Construct a **recursive-descent** parser for the following grammar. 5

$$\begin{aligned} S &\rightarrow \texttt{if } C \texttt{ then } S \\ &\mid \texttt{while } C \texttt{ do } S \\ &\mid \texttt{id = num ;} \\ &\mid \texttt{id ++ ;} \\ C &\rightarrow \texttt{id == num} \mid \texttt{id != num} \end{aligned}$$

A-3/ (a) Consider the following grammar

$$\begin{aligned} S &\rightarrow \text{if } (E) \text{ then } (E)\ X \mid E \\ X &\rightarrow \epsilon \mid \text{else } (E) \\ E &\rightarrow \text{true } Y \mid \text{false } Y \\ Y &\rightarrow \epsilon \mid \text{and } (E) \end{aligned}$$

where **S** is the start symbol and {**if, then, else, and, true, false, (,)**} is the set of terminals. Compute the associated **FIRST** and **FOLLOW** sets. Then construct the **LL(1)** parsing table for the grammar (put only the production numbers in the table). 6

(b) Briefly explain the error recovery strategy in **predictive** parsing. 3

A-4/ (a) Differentiate between scanning and parsing. Name some popular tools used as lexer and parser. 2

(b) The following grammar for `if-then-else` statements is proposed to remedy the dangling else ambiguity 5

$$\begin{aligned} stmt &\rightarrow \texttt{if } exp \texttt{ then } stmt \\ &\mid matched\_stmt \\ matched\_stmt &\rightarrow \texttt{if } exp \texttt{ then } matched\_stmt \texttt{ else } stmt \\ &\mid other \end{aligned}$$

Show that this grammar is still ambiguous.

(c) Explain how a regular expression can be converted into a context-free-grammar by considering the regular expression $(a \mid b)^*abb$. 2

SECTION B

B-1/ (a) What are the different type of conflicts can occur in **shift-reduce** parsing? 2

(b) Consider the following grammars and associated semantic actions.

CamScanner

$G \rightarrow F$	$G.p = F.p$
$F \rightarrow F_1 \wedge F_2$	$F.p = And(F_1.p, F_2.p)$
$F \rightarrow F_1 \vee F_2$	$F.p = Or(F_1.p, F_2.p)$
$F \rightarrow \neg F_1$	$F.p = Neg(F_1.p)$
$F \rightarrow F_1 \Rightarrow F_2$	$F.p = Or(Not(F_1.p), F_2.p)$
$F \rightarrow (F_1)$	$F.p = F_1.p$
$F \rightarrow id$	$F.p = id.lexeme$

i. Say whether each attribute of a non-terminal is inherited or synthesized and why. 2

ii. Show the annotated parse tree for the expression $\neg(A \wedge (A \Rightarrow B))$ 2

iii. Write a **yacc/bison** program to generate a parser for the given grammar. 3

B-2/ (a) Translated the expression $-(a+b)*(c+d)+(a+b+c)$ into i) quadraples, ii) triples, iii) indirect triples. 3

(b) Consider the following code segments that count the number of primes from **2** to n using the *sieve* method on a suitably large array a. Translate the program in to three address statements. Assume integers require **4** bytes. 4

```
for(i=2; i<=n; i++)
    a[i] = TRUE;
count = 0;
s = sqrt(n);
for(i=2; i<=s; i++)
    if(a[i])  /* i has been found to be a prime*/
      count++;
for(j=2*i; j<=n; j = j+1)
     a[j] = FALSE; /* no multiple of i is a prime*/
```

(c) Define copy propagation. What is the advantage of copy propagation? 2

B-3/ (a) Upon determining the basic blocks, construct a control-flow graph corresponding to the following three-address code. 2.5

(1) $i = m - 1$
(2) $j = n$
(3) $t1 = 4 * n$
(4) $v = a[t1]$
(5) $i = i + 1$
(6) $t_2 = 4 * i$
(7) $t_3 = a[t_2]$
(8) if $(t_3 < v)$ goto **(5)**
(9) $j = j - 1$
(10) $t_4 = 4 * j$
(11) $t_5 = a[t_4]$
(12) if $(t_5 > v)$ goto **(9)**
(13) if $(i >= j)$ goto **(23)**
(14) $t_6 = 4 * i$
(15) $x = a[t_6]$

(16) $t_7 = 4 * i$
(17) $t_8 = 4 * j$
(18) $t_9 = a[t_8]$
(19) $a[t_7] = t_9$
(20) $t_{10} = 4 * j$
(21) $a[t_{10}] = x$
(22) goto(5)
(23) $t_{11} = 4 * i$
(24) $x = a[t_{11}]$
(25) $t_{12} = 4 * i$
(26) $t_{13} = 4 * n$
(27) $t_{14} = a[t_{13}]$
(28) $a[t_{12}] = t_{14}$
(29) $t_{15} = 4 * n$
(30) $a[t_{15}] = x$

(b) What are the objectives of code optimization? Apply, in stages, all valid and relevant optimizing transformations to the flow graph obtained in B-3(a) 5

(c) Construct the DAG for the basic block 1.5

(1) $d = b * c$
(2) $e = a + b$
(3) $b = b * c$
(4) $a = e - d$

B-4/ (a) Consider the following *C* code that compute **Fibonacci** numbers recursively. Suppose that the activation record for f includes the following elements in order: *(return value, argument* n*, local* s*, local* t*)* Assume the initial call is $f(5)$

```
int f(int n) {
      int t, s;
      if(n < 2) return 1;
      s = f(n-1);
      t = f(n-2);
      return s+t;
}
```

i. Show the complete activation tree. 3

ii. What does the stack and its activation records look like the first time $f(1)$ is about to return? [*hint. snapshot*] 3

(b) Differentiate between static and dynamic storage allocation. What is the role of the garbage collector? 3

CamScanner

Department of Computer Science and Engineering
University of Chittagong
7th Semester B.Sc. Engineering Exam 2022
CSE 711: COMPILERS
Marks: 52.5 Time: 4 hours

[Answer any three of the four questions from each of the two sections below. Answers to parts of a question should appear contiguously in the answer script. Ensure readability of your handwriting.]

SECTION A

A-1/ (a) Certain aspects of computer architecture design relate to elements of compiler design and implementation in significant ways. Identify them and briefly explain the connection.

(b) What tasks are performed in the *front-end* and *back-end* of a compiler, and why? How are the *phases* related to the *passes*?

(c) What advantages are there to a language-processing system in which the compiler produces assembly language rather than machine language?

(d) *Symbol table is a necessary component of a compiler* – justify this statement with an example. [2+3+2+2]

A-2/ (a) Briefly explain the error recovery strategies in syntax analysis.

(b) Introduce the notions of *token*, *pattern* and *lexeme*. Point out how they are related.

(c) Describe the language denoted by the following regular expression and provide transition diagrams to recognize the language: **(a|b)*abb (a|b)**

(d) How does the lexical analyzer read the input string and break it into lexemes. [3+2+3+1]

A-3/ (a) Let $\mathbb{G}$ be a Context Free Grammar for which the production rules are given below:

$$S \longrightarrow aB \mid bA$$
$$A \longrightarrow a \mid aS \mid bAA$$
$$B \longrightarrow b \mid bS \mid aBB$$

Derive the string **aaabbabbba** using the above grammar (using Left Most Derivation and Right Most Derivation).

(b) Why is left recursion not a problem for bottom-up parsing?

(c) Consider the following grammar and eliminate left recursion

$$X \longrightarrow XSb \mid Sa \mid b \qquad S \longrightarrow Sb \mid Xa \mid a$$

(d) Left factor the following grammar: [2+1+4+2]

$$A \longrightarrow ABd \mid Aa \mid a \qquad B \longrightarrow Be \mid b$$

A-4/ (a) How does recursive descent parser work? Describe with an example.

(b) Compute the FIRST and FOLLOW for the given CFG:

$$S \longrightarrow aXZh \qquad X \longrightarrow cY \qquad Y \longrightarrow bY \mid \epsilon$$
$$Z \longrightarrow EF \qquad E \longrightarrow g \mid \epsilon \qquad F \longrightarrow f \mid \epsilon$$

(c) For the grammar below, a partial **LL(1)** parsing table is presented along with the grammar. Entries that need to be filled are indicated as $\mathbf{E_1}$, $\mathbf{E_2}$, and $\mathbf{E_3}$. [3+3+3]

$S \longrightarrow aAbB \mid bAaB \mid \epsilon$
$A \longrightarrow S$
$B \longrightarrow S$

	a	b	$
S	$\mathbf{E_1}$	$\mathbf{E_2}$	$S \rightarrow \epsilon$
A	$A \rightarrow S$	$A \rightarrow S$	error
B	$B \rightarrow S$	$B \rightarrow S$	$\mathbf{E_3}$

SECTION B

B-1/ (a) What is the difference between LR parser and LL parser? Explain your answer.

(b) Verify that following grammar is $LL(1)$ or not?

$$S \longrightarrow Ab\$ \quad A \longrightarrow (bAb) \quad A \longrightarrow (Ab) \quad A \longrightarrow \lambda$$

(c) On input **(id)** $*$ **id**, give the sequence of stack and input contents for the following parsing table.

Productions	
1	$T \longrightarrow F$
2	$T \longrightarrow T * F$
3	$F \longrightarrow id$
4	$F \longrightarrow (T)$

	Action					GoTo	
	*	(	)	id	$	T	F
0		S5		S8		2	1
1	R1	R1	R1	R1	R1		
2	S3				Acc		
3		S5		S8			4
4	R2	R2	R2	R2	R2		
5		S5		S8		6	1
6	S3		S7				
7	R4	R4	R4	R4	R4		
4	R3	R3	R3	R3	R3		

[2+4+3]

B-2/ (a) Construct an LR parsing table for the given context-free grammar:

$$S \longrightarrow AA \qquad A \longrightarrow aA \mid b$$

(b) Differentiate between inherited and synthesized attributes with examples.

(c) Draw a dependency graph for the expression: $\mathbf{3 * 5 * 2}$. Production rules are given below:

Production	Semantic Rules
1) $T \longrightarrow FT'$	$T'.inh = F.val$ $T.val = T'.syn$
2) $T' \longrightarrow *FT'_1$	$T'_1.inh = T'.inh \times F.val$ $T'.syn = T'_1.syn$
3) $T' \longrightarrow \epsilon$	$T'.syn = T'.inh$
4) $F \longrightarrow$ **digit**	$F.val =$ **digit**$.lexval$

[5+2+2]

B-3/ (a) What is the difference between L-attributed and S-attributed grammar? Is the following grammar L-attributed or not? Explain your answer.

PRODUCTION	SEMANTIC RULES
$A \longrightarrow BC$	$A.s = B.b$ $B.i = f(C.c, A.s)$

(b) Generate code for the following three-address statements assuming **a** and **b** are arrays whose elements are 4-byte values: $x = a[i] \quad y = b[i] \quad z = x * y$

(c) Write quadruples, triples and indirect triples for the expression: [3+3+3]

$$-(\mathbf{a} * \mathbf{b}) + (\mathbf{c} + \mathbf{d}) - (\mathbf{a} + \mathbf{b} + \mathbf{c} + \mathbf{d})$$

B-4/ (a) Consider the following program in 3-address intermediate code

(1) $a = 2$
(2) $b = 3$
(3) $t = a$
(4) $r_1 = 0$
(5) if $(t <= 0)$ goto (9)
(6) $t_1 = 2 * a$
(7) $r_1 = t_1 + r_1$
(8) goto (5)
(9) $t = b$
(10) $r_2 = 0$
(11) if $(t <= 0)$ goto (15)
(12) $t_2 = 7 * b$
(13) $r_2 = t_2 + r_2$
(14) goto (11)
(15) $t_3 = r_1 + r_2$
(16) $r = 8 * t_3$

i. Indicate where new basic blocks start. For each basic block, give the line number such that the instruction in the line is the first one of that block.

ii. Give names B_1, B_2, ... for the program?s basic blocks in the order the blocks appear in the given listing. Draw the control flow graph making use of those names. Don't put in the code into the nodes of the flow graph, the labels B_i are good enough.

(b) What is code optimization? Explain machine dependent and independent code optimization. [3+3+3]

PARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY CHITTAGONG
7TH SEMESTER BSC ENGINEERING EXAM 2021

CSE 711 (*Compilers*)
MARKS: 52.5 TIME: 4 Hrs

Using section-wise separate answer-scripts, answer any three of the four questions from each of the following two sections. Write legibly.

SECTION A

Question 1: (3+8+1)+1.75 marks

(a) Consider the following grammar, with terminal symbols *let*, *in*, *ident*, :=, and $:

$$T \rightarrow S\$$$
$$S \rightarrow D\ in\ E$$
$$D \rightarrow V\ D \mid \epsilon$$
$$V \rightarrow ident := E \mid let\ ident := E$$
$$E \rightarrow ident \mid \epsilon$$

Now answer the followings:

i. Write the FIRST and FOLLOW sets for the grammar.
ii. Compute the LL(1) parse table for the grammar.
iii. Is the grammar LL(1)? Why or why not?

(b) Define and differentiate between compile time errors and runtime.

Question 2: 2+(1+2+2+1.75) marks

(a) How does a parser plays an important role in a compiler design?

(b) The following is a grammar for regular expressions over symbols a and b only, using + in place of | for union, to avoid conflict with the use of vertical bas as a metasymbol in grammars:

$$rexpr \rightarrow rexpr\ +\ rterm \mid rterm$$
$$rterm \rightarrow rterm\ \ rfactor \mid rfactor$$
$$rfactor \rightarrow rfactor\ * \mid rprimary$$
$$rprimary \rightarrow a \mid b$$

Now answer the followings:

i. Left factor this grammar.
ii. Does left factoring make the grammar suitable for top-down parsing?
iii. In addition to left factoring, eliminate left recursion from the original grammar.
iv. Is the resulting grammar suitable for top-down parsing?

Question 3: 3+3+2.75 marks

(a) Introduce the notions of *token*, *pattern* and *lexeme*. Point out how they are related.

(b) Pertaining to the principles of programming languages, clarify the following two distinctions: i) *environment* and *state*, ii) *dynamic* and *static*.

(c) What is the function of lexical analyzer? Explain the reason for separating the analysis phase of compiling into lexical analysis and parsing.

Question 4: 2.75+3+3 marks

Consider the following grammar

$$S \rightarrow S\ a \mid b \mid error\ a$$

Now answer the followings.

(a) Construct the SLR sets of items for the grammar.

(b) Show the PDA or transition table for recognizing viable prefixes of this grammar.

(c) Compute GOTO function for these sets of items.

SECTION B

Question 5: 2+2.75+2+2 marks

(a) Define synthesized attribute and inherited attribute with illustrative examples.

(b) Construct a syntax-directed translation scheme that translates arithmetic expressions from postfix notation into infix notation.

(c) Give annotated parse trees for the inputs $9 - 5 + 2$ and $9 - 5 * 2$.

(d) What are S-attributed and L-attributed grammars?

(b) Discuss the process of input

B-4/ (a) How would you

Question 6: 4.75+4 marks

(a) Consider the following grammar and associated parsing table.

Show the execution of the parser on the string *abbc*.

(b) Translate the arithmetic expression $a * -(b + c)$ into i) a syntax tree and ii) three address code

(c) What is the intermediate code representation for the following expression: *a or b and not c*

```
0: S -> X
1: X -> a X c
2: X -> X X
3: X -> b
```

State	a	b	c	$	S	X
1	s2	s3			g5	
2	s2	s3				g7
3	r3	r3	r3	r3		
4	s2	s3		r0		
5				Acpt		
6		s3	s8			g7
7	r2	r2	r2	r2		
8	r1	r1	r1	r1		

Question 7: 3+3+2.75 marks

(a) Reconstruct the C function that got compiled into the following x86-64 assembly.

```
fA:
movq    (%rdi), %rax
cmpq    %rsi, %rax
jl      .L1
testq   %rax, %rax
jne     .L3
.L1:
ret
.L3:
movq    %rsi, (%rdi)
jmp     .L1
```

The function takes two arguments that are passed through `%rdi` and `%rsi` respectively, as is the convention, and likewise for the return value.

(b) For the (not-very-meaningful-but-syntactically-correct) C function presented, produce its operationally equivalent C code that is free from compound logical operations (i.e., no `||` or `&&`) and does not use any of the following keywords: `else, for, do, while`.

```
long fn(long p, long q, long r, long n)
{
long v;
if(p <0 && q<0)
v=0;
else if (p+q >= r)
v = p+q;
else v = (p>q)? p : q;

for(int i=1; i<=n; i++)
{
if(p > q+i || q > p+i)
v += (p-i)*(q-i);
}
return v;
}
```

(c) What is the role of *stack frames* (a.k.a. *activation records* or *call stack*)? How do the local variables of a function get represented in the associated frame? What are *caller* saved and *callee* saved registers?

Question 8: 1.75+3+4 marks

(a) Why compilers use intermediate representations instead of translating directly from source to target language?

(b) Draw the control flow graph for the following program:

```
test:
  cjmp (x < 0) done
  x = x - y
  y = y + 1
  jump test
done:
  z = y
```

(c) Given the following section of a three-address code for the bubble-sort algorithm, construct the associated flow-graph. Apply all the applicable optimizations techniques and show the evolution of the basic block through several optimization stages.

$$
\begin{array}{ll}
t_8 & \leftarrow j-1 \\
t_9 & \leftarrow 4*t_8 \\
temp & \leftarrow A[t_9] \\
t_{10} & \leftarrow j+1 \\
t_{11} & \leftarrow t_{10}-1 \\
t_{12} & \leftarrow 4*t_{11} \\
t_{13} & \leftarrow A[t_{12}] \\
t_{14} & \leftarrow j-1 \\
t_{15} & \leftarrow 4*t_{14} \\
A[t_{15}] & \leftarrow t_{13} \\
t_{16} & \leftarrow j+1 \\
t_{17} & \leftarrow t_{16}-1 \\
t_{18} & \leftarrow 4*t_{17} \\
A[t_{18}] & \leftarrow temp
\end{array}
$$

University of Chittagong
Department of Computer Science and Engineering
7^{th} Semester B.Sc. Engineering Examination 2020
CSE 711: Compilers
Marks: 52.5 Time: 4:00 Hours

[The figures in the right hand margin indicate full marks. Answer any **Three** questions from **Section-A** and any **Three** questions from **Section-B**]

Section - A

1. (a) What tasks are performed in the *front-end* and *back-end* of a compiler, and why? How are the *phases* related to the *passes*? (4)

   (b) How can immediate left-recursion be eliminated? Illustrate how left-recursion involving derivation of multiple steps can be eliminated using the following grammar: (3)

$$S \rightarrow Pa \mid q$$
$$P \rightarrow Pc \mid Sd \mid \epsilon$$

   (c) Introduce the error recovery strategies in syntax analysis. (1.75)

2. Consider the following grammar:

$$S \rightarrow a\,S\,b\,S \mid b\,S\,a\,S \mid \epsilon$$

   (a) Compute FIRST and FOLLOW. (3)

   (b) Construct a predictive parsing table. (4)

   (c) Is the grammar LL(1)? (1.75)

3. (a) Consider the following grammar:

$$S \rightarrow (\ L\ ) \mid id$$
$$L \rightarrow L\ ,\ S \mid S$$

      i. Construct the LR(0) items. (1.75)

      ii. Compute the LR(0) states. (5)

   (b) Differentiate between synthesized and inherited attributes. (2)

4. (a) Write a Yacc/Bison program that takes boolean expressions as input using following grammar and produces the truth value of the expressions. (5)

```
bexpr -> bexpr or bterm | bterm
bterm -> bterm and bfactor | bfactor
bfactor -> not bfactor | ( bexpr ) | true | false
```

   (b) What is the role of symbol table in compilers? Remark briefly on the key technical aspects in creating and managing the *symbol table*. (3.75)

Please go on to the next page...

Section - B

5. (a) When can a compiler's implementation of a language be called *strongly typed*? What is *widening* and *narrowing* in type conversion? (2)

   (b) Given an LR(k) item $[A \rightarrow \alpha \,.\, \gamma, \delta]$ from a production $A \rightarrow \beta$, describe what A, α, γ and δ signify. (3)

   (c) Employing the following Action/Goto table (3.75)

State	Action		Goto	
	c	$	S	T
0	Shift 1	Reduce $T \rightarrow \epsilon$	2	3
1	Shift 3	Reduce $S \rightarrow c$	0	
2		Accept	3	0
3	Shift 1	Accept		1

   demonstrate the operation of a shift-reduce parser with the string "**c**". Show the stack contents, input and action at each step (assuming state 0 initially).

6. (a) Consider the following program in 3-address intermediate code

```
(1)  a = 2
(2)  b = 3
(3)  t = a
(4)  r1 = 0
(5)  if (t ≤ 0) goto (9)
(6)  t1 = 2 * a
(7)  r1 = t1 + r1
(8)  goto (5)
(9)  t = b
(10) r2 = 0
(11) if (t ≤ 0) goto (15)
(12) t2 = 7 * b
(13) r2 = t2 + r2
(14) goto (11)
(15) t3 = r1 + r2
(16) r = 8 * t3
```

```
(1)  t8 = j - 1
(2)  t9 = 4 * t8
(3)  tmp = A[t9]
(4)  t10 = j + 1
(5)  t11 = t10 - 1
(6)  t12 = 4 * t11
(7)  t13 = A[t12]
(8)  t14 = j - 1
(9)  t15 = 4 * t14
(10) A[t15] = t13
(11) t16 = j + 1
(12) t17 = t16 - 1
(13) t18 = 4 * t17
(14) A[t18] = tmp
```

   i. Indicate where new basic blocks start. For each basic block, give the line number such that the instruction in the line is the first one of that block. (3)

   ii. Give names $B_1, B_2, \ldots$ for the program's basic blocks in the order the blocks appear in the given listing. Draw the control flow graph making use of those names. (3.75)

   (b) Construct the DAG for the basic block (2)

```
(1) d = b * c
(2) e = a + b
(3) b = b * c
(4) a = e - d
```

7. (a) What is the difference between a parse tree and an abstract syntax tree (AST)? (2)

   (b) Introduce the idea of semantics preserving (local) optimizations in code optimization. (3)

   (c) Illustrate how *triples*, *quadruples* and *static single assignment* (**SSA**) form are different in representing intermediate codes. (3.75)

8. Consider the following merge-sort program.

```
void mergesort(int a[], int i, int j)
{
    int mid;
    if (i < j)
    {
    mid = (i+j)/2;
    mergesort(a, i, mid);  // left recursion
    mergesort(a, mid+1, j);  // right recursion
    merge(a, i, mid, j);  // merging of two sorted sub-arrays
    }
}
void merge(int a[], int p, int q, int r)
{
        /* Combine the elements back in a[p . . r] by merging

        the two sorted subarrays a[p . . q] and

        a[q+1 . . r] into a sorted sequence. */

        . . .
}

main()
{
        int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
        mergesort(a, 0, 10);
}
```

(a) *Define following terms:* activation tree and activation record. (2)

(b) *Show the complete activation tree* for the given *merge-sort* program. (3)

(c) *Draw the activation records* for the given *merge-sort* program. (3.75)

University of Chittagong
Department of Computer Science and Engineering
7th Semester B.Sc. (Engineering) Examination 2018
Course Title: Compilers
Course Code: CSE 711, Full Marks: 52.5, Duration: 4:00 Hours

Answer any **three** questions from each section.
[All parts of a question should be answered sequentially; figures in the right hand margin indicate full marks]

Section A

1. a) Even though grammars can do what regular expressions do, why is regular expression employed instead of grammar in lexical analysis? 2

   b) For each of the followings, fill-in the blanks with succinct and most relevant clauses 9X0.75
   i) A phase of a compiler represents .....whereas a pass indicates.....
   ii) A.....is called inherently ambiguous when.....
   iii) A static policy in a language denotes.....whereas a dynamic policy refers to.....
   iv) The use cases of compiler technologies beyond designing a compiler include .....
   v) A pair of input buffers, rather than a single buffer, helps scanning because.....
   vi) One of the benefits in factoring a compiler into front-end and back-end is.....
   vii) A symbol table is.....
   viii) An implementation of a language is strongly typed if.....
   ix) An abstract syntax tree is.....

2. a) What are the common categories of programming errors? Explain how *lexical analyzer generator (f)lex* resolves conflict when several prefixes of the input match one or more patterns. 2.75

   b) Consider the following two grammars 3+3

$A \rightarrow a$	$A \rightarrow Aa$
$A \rightarrow aA$	$A \rightarrow a$
(G_1)	(G_2)

   Identify the issues with these two grammars when predictive parsing is at tempted. Then determine and apply the transformations to make each of them fit for the purpose.

3. a) Consider the following grammar and corresponding ACTION and GOTO table: 4.75

   1. $S' \rightarrow S$
   2. $S \rightarrow AB$
   3. $A \rightarrow aA$
   4. $A \rightarrow a$
   5. $B \rightarrow bb$

State	ACTION			GOTO		
	a	b	$	S	A	B
0	s5			1	2	
1			*accept*			
2		s3				4
3		s7				
4			r2			
5	s5	r4			6	
6		r3				
7			r5			

   Show the parse for string aaabb with the state/content of the stack, (remaining) input and the action taken in each step shown in tabular form.

   b) Construct the LR (0) automaton (preferably, DFA) with the appropriate sets of items produced from the following (augmented) grammar. 4

   $S' \rightarrow S$
   $S \rightarrow Sab$

   Where {a, b} is the set of terminals.

4. a) What is a sentinel? Describe the position of pointers at various phases during lexical analysis using sentinels. 2.75

   b) Define tokens, patterns, and lexemes. What are the different levels of syntax error handler? 3

c) Consider the grammar:

$$S \rightarrow (L) \mid a$$
$$L \rightarrow L, S \mid S$$

i) What are the terminals, non terminals, and start symbol?
ii) Find parse tree for the following sentence:
(a, ((a,a), (a,a)))
iii) Construct a leftmost derivation for the sentence in (ii).

Section B

5. a) Given an LR (1) item [A→α. γ, δ], what lookahead allows a shift to be performed (ignoring any other LR (1) items added to the state by closure)? Relatedly, state the reason why there is no shift-shift conflict in shift-reduce parsing. 2.75

b) Introduce Chomsky hierarchy of languages/grammars. In this connection, recall that there are aspects in programming language processing where context sensitivity is involved (e.g, declaration of variables required before their use in C). How does compiler design incorporate such a characteristic even though the categories of the grammars do not go beyond CFGs? 3

c) Briefly illustrate how CYK algorithm works. How practical would it be to do parsing using CYK algorithm? 3

6. a) Show how short-circuit code (in intermediate representation) can be generated for a statement involving compound Boolean expression like the following 2

If (a>70 && (b<50 || c>30)) z = 0;

b) Illustrate how a (*do-while, for or while)* loop gets translated into three address code. 3

c) How does a dependency graph take shape? How and when can attributes be computed from a dependency graph? Define synthetic and inherited attributes, and exemplify using a suitable attribute grammar. 3.75

7. a) Calculate the instruction cost of the given instruction sequence 2

MOV b, R0
ADD c, R0
MOV R0, a.

b) Discuss about the followings: 4.5

i) Constant folding
ii) dead-code elimination and
iii) code motion

c) Define live variable. Given the primary structure preserving transformations on basic blocks. 2.25

8. a) What measures are typically used to assess the performance of an automated garbage collector? Describe how counting references to objects makes garbage collection possible. 2.75

b) Explain how procedure calls get managed during the runtime of a program. What information must an activation record contain? 3

c) What are the typical functional partitions of the memory in which a running program resides? Distinguish the static and dynamic portions in this regard. Why/how does caching affect program performance? 3