Sample Question Format
(For all courses having end semester Full Mark=50)

KIIT Deemed to be University
Online End Semester Examination(Autumn Semester-2021)

Subject Name & Code: **Compiler Design (CS3008)**
Applicable to Courses:

Full Marks=50
Time:2 Hours

SECTION-A(Answer All Questions. Each question carries 2 Marks)

Time:30 Minutes
(7×2=14 Marks)

Question No	Question Type (MCQ/SAT)	Question	CO Mapping	Answer Key (For MCQ Questions only)
Q.No:1		LR stands for: (a) Left to Right Input Scanning (b) Left to Right Input Scanning and Right Most Derivation in Reverse (c) Left to Right Input Scanning and Right Most Derivation (d) None of the above	2	(b)
		The LR(0) item B a·β refers to: (a) The parser has completed the parsing (b) The parser is about start parsing the derivatives of β (c) The parser has parsed *a* and is about start parsing the derivatives of β (d) The parser is about start parsing the derivatives of *a*	2	(c)
		Consider a grammar A $X_1X_2X_3X_4$. If, X_1 is a non-terminal and X_2 produces λ,	2	(c)

<table>
<tr><td></td><td></td><td>then FOLLOW(X_1) is:
(a) FIRST(X_1)
(b) FIRST(X_2)
(c) FIRST(X_3)
(d) FIRST(X_4)</td><td></td><td></td></tr>
<tr><td></td><td></td><td>Match all items in Group 1 with correct options from those given in Group 2.
<table><tr><td>Group 1</td><td>Group 2</td></tr><tr><td>P.Regular expression</td><td>1.Syntax analysis</td></tr><tr><td>Q.Pushdown automata</td><td>2.Code generation</td></tr><tr><td>R.Dataflow analysis</td><td>3.Lexical analysis</td></tr><tr><td>S.Register allocation</td><td>4.Code optimization</td></tr></table>
(a) P-4. Q-1, R-2, S-3
(b) P-3, Q-1, R-4, S-2
(c) P-3, Q-4, R-1, S-2
(d) P-2, Q-1, R-4, S-3</td><td>2</td><td>(b)</td></tr>
<tr><td><u>Q.No:2</u></td><td></td><td>An LALR(1) parser for a grammar G can have shift-reduce (S-R) conflicts if and only if
<u>(a)</u> The SLR(1) parser for G has S-R conflicts
<u>(b)</u> The LR(1) parser for G has S-R conflicts
<u>(c)</u> The LR(0) parser for G has S-R conflicts
<u>(d)</u> The LALR(1) parser for G has reduce-reduce conflicts</td><td>2</td><td>(b)</td></tr>
<tr><td></td><td></td><td>Consider the grammar defined by the following production rules, with two operators ∗ and +
S T * P
T U | T * U
P Q + P | Q
Q Id
U Id
Which one of the following is TRUE?
(a) + is left associative, while ∗ is right associative
(b) + is right associative, while ∗ is left associative
(c) Both + and ∗ are right associative
(d) Both + and ∗ are left associative</td><td>2</td><td>(b)</td></tr>
</table>

		Consider the following grammar. S -> S * E S -> E E -> F + E E -> F F -> id Consider the following LR(0) items corresponding to the grammar above. (i) S -> S * .E (ii) E -> F. + E (iii) E -> F + .E Given the items above, which two of them will appear in the same set in the canonical sets-of-items for the grammar? (a) (i) and (ii) (b) (ii) and (iii) (c) (i) and (iii) (d) None of the above	2	(d)
		Consider the following grammar: S → FR R → S | ε F → id In the predictive parser table, M, of the grammar the entries M[S, id] and M[R, $] respectively. (a) {S → FR} and {R → ε } (b) {S → FR} and { } (c) {S → FR} and {R → *S} (d) {F → id} and {R → ε}	2	(a)
Q.No:3		Consider the following translation scheme. S → ER R → *E{print("*");}R | ε E → F + E {print("+");} | F F → (S) | id {print(id.value);} Here id is a token that represents an integer and id.value represents the corresponding integer value. For an input '2 * 3 + 4', this translation scheme prints (a) 2 * 3 + 4 (b) 2 * +3 4	3	(d)

		(c) 2 3 * 4 + **(d)** 2 3 4+*		
		Consider the following expression grammar. The semantic rules for expression calculation are stated next to each grammar production. E → number E.val = number. val | E '+' E E(1).val = E(2).val + E(3).val | E '×' E E(1).val = E(2).val × E(3).val Assume the conflicts in Part (a) of this question are resolved and an LALR(1) parser is generated for parsing arithmetic expressions as per the given grammar. Consider an expression 3 × 2 + 1. What precedence and associativity properties does the generated parser realize? (a) Equal precedence and left associativity; expression is evaluated to 7 (b) Equal precedence and right associativity; expression is evaluated to 9 (c) Precedence of '×' is higher than that of '+', and both operators are left associative; expression is evaluated to 7 (d) Precedence of '+' is higher than that of '×', and both operators are left associative; expression is evaluated to 9	3	(b)
		Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {a, b}}. S ⟶ **aA** { print 1 } S ⟶ **a** { print 2 } A ⟶ **Sb** { print 3 } Using the above SDTS, the output printed by a bottom-up parser, for the input **aab** is (a) 1 3 2	3	(c)

		(b) 2 2 3 (c) 2 3 1 (d) Syntax Error		
		In a bottom-up evaluation of a syntax directed definition, inherited attributes can (a) always be evaluated (b) be evaluated only if the definition is L-attributed (c) be evaluated only if the definition has synthesized attributes (d) never be evaluated	3	(b)
Q.No:4		Some code optimizations are carried out on the intermediate code because (a) they enhance the portability of the compiler to other target processors (b) program analysis is more accurate on intermediate code than on machine code (c) the information from dataflow analysis cannot otherwise be used for optimization **(d)** the information from the front end cannot otherwise be used for optimization	5	(a)

		Consider the following C code segment.
		
		for (i = 0, i<n; i++)
		{
		  for (j=0; j<n; j++)
		  {
		    if (i%2)
		    {
		      x += (4*j + 5*i);
		      y += (7 + 4*j);
		    }
		  }
		}
		Which one of the following is false?
		(a) The code contains loop invariant computation
		(b) There is scope of common sub-expression elimination in this code
		(c) There is scope of strength reduction in this code
		(d) There is scope of dead code elimination in this code
			5	

		Consider the code segment
		
		int i, j, x, y, m, n;
		n=20;
		for (i = 0, i < n; i++)
		{
		for (j = 0; j < n; j++)
		{
		if (i % 2)
		{
		x + = ((4*j) + 5*i);
		y += (7 + 4*j);
		}
		}
		}
		m = x + y;
		
		Which one of the following is false
		(a) The code contains loop invariant computation
		(b) There is scope of common sub-expression elimination in this code
		(c) There is scope of strength reduction in this code
		(d) There is scope of dead code elimination in this code
			5	(d)

		In compiler terminology reduction in strength means (a) Replacing run time computation by compile time computation (b) Removing loop invariant computation (c) Removing common subexpressions **(d)** Replacing a costly operation by a relatively cheaper one	5	(d)
Q.No:5		One of the purposes of using intermediate code in compilers is to (a) make parsing and semantic analysis simpler. (b) improve error recovery and error reporting. (c) increase the chances of reusing the machine-independent code optimizer in other compilers. (d) (D) improve the register allocation.	4	(c)
		In the context of compilers, which of the following is/are	4	(d)

		NOT an intermediate representation of the source program? (a) Three address code (b) Abstract Syntax Tree (AST) (c) Control Flow Graph (CFG) **(d)** Symbol table		
		There are three source languages and four target languages available for the compilation process. How many translation processes are required without using Intermediate Code? (a) 3 (b) 4 (c) 12 **(d)** 7	4	(c)
		Three address code is: (a) High-level Language (b) High-level intermediate language (c) Medium-level intermediate language **(d)** Low-level intermediate language	4	(c)
Q.No:6		The number of tokens in the following C statement is: printf("i = %d, &i = %x", i, &i); (a) 3 (b) 26 (c) 10 (d) 21	1	(c)
		In a compiler, keywords of a language are recognized during (a) parsing of the program (b) the code generation (c) the lexical analysis of the program (d) dataflow analysis	1	(c)
		The lexical analysis for a modern computer language such as Java needs the power of which one of the following machine models in a necessary and sufficient sense? (a) Finite state automata (b) Deterministic pushdown automata (c) Non-Deterministic pushdown automata **(d)** Turing Machine	1	(a)
		Consider the following statements: (I) The output of a lexical	1	(d)

		analyzer is groups of characters. (II) Total number of tokens in printf("i=%d, &i=%x", i, &i); are 11. (III) Symbol table can be implementation by using array and hash table but not tree. Which of the following statement(s) is/are correct? (a) Only (I) (b) Only (II) and (III) (c) All (I), (II), and (III) **(d)** None of these		
Q.No:7		A grammar that is both left and right recursive for a non-terminal is (a) Ambiguous (b) Unambiguous (c) Information is not sufficient to decide whether it is ambiguous or Unambiguous. **(d)** None of the above	2	(c)
		The Basic Block (a) Includes the leader and the next statements till the next leader excluding it (b) Includes the leader and the next statements till the end of the program (c) Includes the leader and the next statements till the next leader including it (d) Includes the leader and the next statement	4	(a)
		Backtracking in TOP_DOWN parser (a) Goes to the previous expansion of rule (b) Goes to the start symbol of the grammar where there is an alternative definition (c) Goes to the scanner and tries with other grammar (d) Goes to the previous expansion of rule where there are alternative definitions	2	(d)
		The function λ-Closure(S) (a) Finds the set of all states reachable from S on an input symbol (b) Finds the set of all states reachable from S on any input symbol	1	(c)

		(c) Finds the set of all states reachable from S on λ input (d) None of these		

SECTION-B(Answer Any Three Questions. Each Question carries 12 Marks)

Time: 1 Hour and 30 Minutes
(3×12=36 Marks)

Question No	Question	CO Mapping (Each question should be from the same CO(s))
Q.No:8	Consider the following Grammar: E E – E | E / E | E % E | id (i) Construct a LR parsing table for the given grammar. (ii) Resolve all conflicts (if any) by assuming all binary operators are right associative (iii) Post resolving the conflicts show the parsing of some string derivable by the given grammar.	2
	Consider the following Grammar: S L = R S R L *R L id R L (i) Construct a LALR(1) parsing table for the given grammar. (ii) Show parsing of some string derivable by the given grammar by using LR parsing algorithm	
	Consider the following Grammar: S Aa | dc | bAc | bDa A d | e D f (i) Construct a LR(i) parsing table for the given grammar. (ii) Show the parsing of some string derivable by the given grammar by using LR parsing algorithm	
Q.No:9	(i) What is SDD?	3

	(ii) Explain the role of SDD. (iii) Differentiate between inherited attribute and synthesized attribute. Give example of each. (iv) Which attribute is mandatory and why? Explain with example.	
	num string.string num string string string bit string bit bit 0 bit 1 Consider the above grammar: (a) Write the translation scheme using both attributes to convert the binary into decimal. (b) Draw a dependency graph for some string derivable by the grammar.	
	num sign list list list bit list bit sign - sign + bit 0 bit 1 Consider the above grammar: (a) Write the translation scheme using both attributes to convert the binary into decimal. (b) Draw a dependency graph for some string derivable by the grammar.	

Q.No:10	Explain the importance of Intermediate Code Generation in compiler design.
	
	int a[10], b[10], dot_prod, i;
	dot_prod = 0;
	for (i=0; i<10; i++)
	dot_prod += a[i]*b[i];
	
	Consider the above code fragment and perform the following operations:
	(i) Generate the Three Address Code for the given code
	(ii) Write the TAC in quadruples
	(iii) Write the TAC in triples
		4

	Explain the importance of Intermediate Code Generation in compiler design.
	
	int a[10], b[10], dot_prod, i;
	int* a1;
	int* b1;
	dot_prod = 0;
	a1 = a;
	b1 = b;	

	for (i=0; i<10; i++)
	  dot_prod += *a1++ * *b1++;
	
	Consider the above code fragment and perform the following operations:
	(i) Generate the Three Address Code for the given code
	(ii) Write the TAC in quadruples
	(iii) Write the TAC in triples	

	Explain the importance of Intermediate Code Generation in compiler design.
	
	int dot_prod(int x[], int y[])
	{
	  int d, i;
	  d = 0;
	  for (i=0; i10; i++)
	    dot_prod += x[i]*y[i];
	}
	
	Consider the above code fragment and perform the following operations:
	(i) Generate the Three Address Code for the given code
	(ii) Write the TAC in quadruples
	(iii) Write the TAC in triples	

Q.No:11	What is code optimization? Explain different types of code optimization w.r.t. machine dependent and machine independent code optimization.
	
	Consider the following code fragment:
	
	void Sort(int arr[], int n)
	{
	  int i, j, min_idx;
	  for (i = 0; i < n-1; i++)
	  {
	    min_idx = i;
	    for (j = i+1; j < n; j++)
	    if (arr[j] < arr[min_idx])
	      {
	    min_idx = j;
	          int temp = arr[min_idx];
	          arr[min_idx] = arr[i];
	          arr[i] = temp;
	      }
	  }
	}
	
	(i) Generate the TAC
	(ii) Optimize the code by considering all applicable optimization techniques
		5

	What is code optimization? Explain different types of code optimization w.r.t.	

machine dependent and machine independent code optimization.

Consider the following code fragment:

```
void Sort(int arr[], int n)
{
  int i, key, j;
  for (i = 1; i < n; i++)
  {
    key = arr[i];
    j = i - 1;
    while (j >= 0 && arr[j] > key)
        {
          arr[j + 1] = arr[j];
          j = j - 1;
        }
        arr[j + 1] = key;
  }
}
```

(i) Generate the TAC
(ii) Optimize the code by considering all applicable optimization techniques

What is code optimization? Explain different types of code optimization w.r.t. machine dependent and machine independent code optimization.

Consider the following code fragment:

```
void Sort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)
      for (j = 0; j < n-i-1; j++)
        if (arr[j] > arr[j+1])
            {
                    int temp = arr[j];
                    arr[j] = arr[j+1];
                    arr[j+1] = temp;
            }
}
```

(i) Generate the TAC
(ii) Optimize the code by considering all applicable optimization techniques