INDIAN INSTITUTE OF TECHNOLOGY KHARAGPUR

Department of Computer Science and Engineering

Sub No: CS31003 | **Subject Name: Compilers** | **3rd Year B. Tech**

Class Test 2

Date: Nov 4, 2024 | **Time: 6:00pm – 7:00pm** | **Maximum Marks: 40**

Roll Number : _____ **Name :** _____

[***Answer all the questions. Take and state suitable assumptions, if needed.*** **]**

1. Consider the following grammar (start symbol: based_num) which generates strings of octal and decimal integers (non-negative only).

based_num	→	basechar num
basechar	→	**o | d**
num	→	num digit | digit
digit	→	**0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**

Here, we plan to develop a Syntax-Directed Definition which can evaluate the numerical values of the generated strings (with appropriate base). For instance, the string **d237** evaluates as 237, whereas the string **o237** evaluates to 159. In order to solve this problem, we define one inherited attribute *base* for the nonterminals **basechar**, **digit**, and **num**, and one synthesized attribute *val* for the nonterminals **num**, **digit**, and **based_num**. For syntactically valid but semantically invalid strings (like **o956**), we use a special value called **ERROR**.

(a) Fill in the blanks in the following table with suitable semantic rules. **[7]**

based_num → basechar num	{ based_num.val = _____
	num._____ = _____ }
basechar → o	{ basechar.base = _____ }
basechar → d	{ basechar.base = 10 }
num → num_1 digit	{ num.val = _____
	_____
	_____
	num1._____ = _____
	digit._____ = _____ }
num → digit	{ num._____ = _____
	digit._____ = _____
digit → 0|1|2|3|4|5|6|7|8|9	{ digit.val = _____
	_____
	_____ }

(b) For the two strings **d297** and **o956**, construct and annotate the parse trees using the above SDD.
[4 + 4]

2. (a) The following grammar generates boolean expressions and conditional statements. Design a syntax-directed translation scheme (SDT) to generate three-address codes for this grammar. Avoid redundant jumps and labels. Do **not** use backpatching. In the grammar, **rel** indicates the relational operators like < and >. Write appropriate actions against the productions of the grammar. **[10]**

P → S {

}

S → **if** (B) S1 **else** S2 {

}

S → **id** = E**;** {

}

E → E1 + E2 {

}

E → **id** {

}

B → E1 **rel** E2 {

}

B → (B) {

}

B → B1 **&&** B2 {

}

B → B1 || B2 {

}

(b) Apply the SDT of Part (a) to translate the following statement to a three-address code. In this process, suitably annotate the relevant parse tree. **[15]**

```
if (a > b && (c < d || p > 100))
      x = 0;
else
      x = 1;
```