Department of Computer Science & Engineering

University of Asia Pacific (UAP)

Program: B.Sc. in Computer Science and Engineering

Final Examination **Spring 2021** **4th Year 2nd Semester**

Course Code: CSE 429 **Course Title: Compiler Design** **Credits: 3**

Full Marks: 120* (Written) **Duration: 2 Hours**

* Total Marks of Final Examination: 150 (Written: 120 + Viva: 30)

Instructions:

1. There are **Four (4)** Questions. Answer all of them. All questions are of equal value. Part marks are shown in the margins.
2. Non-programmable calculators are allowed.

1. Consider the fragment of code:

```
sum = 0;
do
{
    if (student1 < student2)
        number++;
     else
         number--;
     sum++;
} while (sum < 10)
```

a) Calculate the Three Address Code (TAC) for the above-stated code. 5

b) Calculate the Quadruples and Triples from the TAC you got in 1(a). 5+5

c) Now, identify the leaders in the TAC you calculated in 1(a). Also, specify the reasons for choosing a leader. 4+3

d) Analyze the TAC statements you got from 1(a) to find out the basic blocks by using the leaders. Also, construct the flow graph for your code based on 1(a) 4+4

2. **a)** The following is Context-Free Grammar over symbols a and b only. 5*4 =20

REXPR→REXPR+ RTERM | RTERM
RTERM→RTERM RFACTOR a | RTERM RFACTOR b | RTERM RFACTOR
RFACTOR→RFACTOR*RPRIMARY | RPRIMARY
RPRIMARY→a | b

i) Examine if the above-stated grammar has Left Factoring? If yes, eliminate that.

ii) Explain how does the elimination of Left Factoring make the grammar more suitable for top-down parsing?

iii) Examine if the above-stated grammar has Left Recursion? If yes, eliminate that.

iv) Explain how does the elimination of Left Recursion make the grammar more suitable for top-down parsing?

b) What is a recursive descent parser? Consider the following Context-Free Grammar(CFG): 10

Exp→ Ax | Bw
A→ xxC |xxyx
C→yyy
B→ xxD
D→ yyz

For the input string “**xxyyzw**” explain how does the recursive descent parser work?

3. Consider the following Context-Free Grammar(CFG): 10*3 =30

P → S P’
P’ → Ɛ | P
S → if S’| while E S | print E | E
S’→ E S”
S” → then T
T → S T’
T’ → else S
E → id | int

And the parsing table is stated below:

i) Calculate first and follow function for the above-stated grammar.

ii) Calculate the LL(1) parsing table for the above-stated grammar.

iii) Can we give an example of a string that can be successfully parsed by the above-stated grammar. Explain the intermediate procedure.

4. Consider the following Context-Free Grammar(CFG): 10*3 =30

S→ i e t S | i e t S e S | w e do S | b L e | s
L → L; S | S

i) Calculate the LR(0) itemsets.

ii) Apply the LR(0) algorithm and find out the parsing table for the above-stated grammar.

iii) Examine the parsing table you got from 4(ii). Can you successfully parse a string using this parsing table? If your answer is no, then please specify your reason.

OR

Apply canonical LR(1) parsing algorithm on the below stated Context-Free Grammar (CFG): 30

G → P | PG
P → id : R
R → ε| id R

i) Calculate the canonical item set of LR(1).

ii) Apply the CLR(1) algorithm and find out the parsing table for the above-stated grammar.

iii) Examine the parsing table you got from 4(ii). Can you successfully parse a string using this parsing table? If your answer is no, then please specify your reason.