Department of Computer Science & Engineering

University of Asia Pacific (UAP)

Program: B.Sc. in Computer Science and Engineering

Final Examination **Fall 2020** **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. **a)** Draw the Abstract Syntax Tree (AST) of the following equation: 8

a:= (b + c) + (d+e) * (-e) * (-e) * (b + c)

b) Now, draw the Directed Acyclic Graph (DAG) of the equation provided in 1(a). 7

c) Now, convert the question of 1(a) DAG to a Three-Address Code (TAC). Draw the Quadruples and Triples of the three-address code. 15

2. 20 + 10 = 30

```
1)  i = 0
2)  if i>= n    goto(18)
3)  j = 0
4)  if j >= n goto(11)
5)  t1 = n * i
6)  t2 = t1 + j
7)  t3 = t2 * 8
8)  c[t3] = 0.0
9)  j = j + 1
10)  goto(4)
11)  i = i + 1
12)  goto(2)
13)  k= 0
14)  t1 = t1 + 10
15)  t2 = t1 * 5
16)  t3 = t2 - 2
17)  c[t2] = 25
18)  if i>= n
             goto(44)
19)  j = 0
20)  if j >= n
             goto(42)
21)  k = 0
22)  if k >= n
             goto(40)
23)  t4 = n * i
24)  t5 = t4 + j
25)  t6 = t5 * 8
26)  t7 = c[t6]
27)  t8 = n * i
28)  t9 = t8 + k
29)  t10 = t9 * 8
30)  t11 = a[t10]
31)  t12 = n * i
32)  t13 = t12 + j
33)  t14 = t13 *8
34)  t15 = b[t14]
35)  t16 = t11 *t15
36)  t17 = t7 + t16
37)  c[t6] = t17
38)  k = k + 1
39)  goto(22)
40)  j = j + 1
41)  goto(20)
```

i) Identify the leaders first and then analyze these statements to find out the basic blocks by using the leaders.
ii) Construct the flow graph for your code from (i).

3. LL(1) parsing table is given below: 30

Non Terminals	**id**	**+**	*****	**(**	**)**	**$**
E	E→TE′			E→TE′		
E′		E′ →+TE′			E′ →Ɛ	E′ →Ɛ
T	T→FT′			T→FT′		
T′		T′→Ɛ	T′→*FT′		T′→Ɛ	T′→Ɛ
F	F → id			F → (E)		

For the grammar:

E→TE′
E′ →+TE′ | Ɛ
T→FT′
T′→*FT′ | Ɛ
F → (E) | id

How will the input string ((id+id)*id)+id will be parsed using the above parsing table?

4. **a)** Consider the following Context Free Grammar(CFG): 10+ 10+ 5+ =25

stmt → if e then stmt
| if e then stmt else stmt
| while e do stmt
| begin list end
| s
list → list; stmt
| stmt

This grammar contains certain statements that are declared as “stmt”. Also, e and s are terminals standing for “conditional expressions” and “other statements” respectively. Here, “**stmt**” and “**list**” are the two Non-Terminals and the rests refer to the Terminals.

i) Calculate the LR(0) itemsets at first.
ii) Then apply the LR(0) algorithm and find out the parsing table for the above stated grammar.
iii) Can we successfully parse a string using the parsing table? If your answer is no, then please specify your reason.

b) Does left recursion/left factoring create any problem while implementing LR parsers? 5

OR

a) Apply canonical LR(1) parsing algorithm on the below stated Context Free Grammar (CFG): 10+ 10+ 5 =25

S → Y| YS

Y → num : X

X → num X

For this reason, you have to calculate the canonical item set and parsing table both. Specify if the grammar can be successfully parsed using LR(1) or not. Give proper reasons behind your choice.

b) There are different types of conflicts that occur while constructing the LR parsing table. Build two different scenarios using those different conflicts. 5