BITS, Pilani- K K Birla Goa Campus Date: 09/05/2019

Course: CS F363 **Semester**-II, 2018-19**, End Semester Examination Marks: 80** Time: 2:00-5:00

Compiler construction **Weightage:** 40% **Mode:** Closed Book Duration: 180 min

Note: No marks will be given if the justification for your answer is not provided.

1. Consider the following grammar
   $S \rightarrow aSbS$
   $S \rightarrow a$
   (a) Construct LR(0) items. $\boxed{5}$
   (b) Construct DFA using LR(0) items. $\boxed{5}$
   (c) Construct LR(0) parsing table. $\boxed{5}$

2. Consider the following grammar.
   S → AA
   A → aA | b

   (a) Construct items for SLR (1) parsing technique. $\boxed{5}$
   (b) Construct DFA using SLR(1) items. $\boxed{5}$
   (c) Construct parsing table for SLR(1). $\boxed{5}$

3. Consider the following grammar.

   $S \rightarrow CC$
   $C \rightarrow Cc \mid d$
   (a) Construct LR(1) items. $\boxed{5}$
   (b) Construct FSM using LR(1) items. $\boxed{5}$
   (c) Construct parsing table using CLR(1) parsing technique. $\boxed{5}$
   (d) Write the pseducode of Construction of the sets of LR(1) items where **Input:** An augmented grammar G'. **Output:** The sets of LR(1) items that are the set of items valid for one or more viable prefixes of G'. $\boxed{5}$

4. Consider the grammar:
   $N \rightarrow 0 \mid num\ L$, $L \rightarrow num\ L \mid D$, $D \rightarrow 11$, Where $V = \{N, L, D\}$, $T = \{num, 11\}$ and N is the start symbol. Note that "," is neither a variable nor a terminal. Assume that the scanner can recognize the final double one in an input string (with two characters of lookahead) and returns the token `11`. `num` can be any value from 0 to 7.
   (a) Augment this as an attribute grammar with semantic rules such that it computes the decimal value of the octal number during parsing. Choose your own attributes, however, the number of attributes must be *minimum* AND the start symbol N must contain the final decimal value of the octal number. $\boxed{8}$
   (b) Draw the parse tree for the number `7311` and show all attributes with their values. $\boxed{2}$

5. Figure 1(a) represents a simple high level language code and Figure 1(b) represents the corresponding 3-address code. Note that using the comment `gcc -fdump-tree-cfg` Figure 1(b) has been generated from the Figure 1(a). Consider the following algorithm which is one of the module for construction of a bipartite graph (for example Petri net) from the Figure 1(b). However, some steps are missing in

Algorithm 1. Fill in the blanks in the Algorithm 1. **Note:** Do not consider any temporary variables in Algorithm 1. **20**

```
int i=1,j=1,k;
#pragma scop
while (i<=10)
   i++;
while (j<=10)
   j++;
#pragma scop
 k=i+j;
return k;
```

(a)

```
int k,j,i;
<bb2>:  i = 1;j = 1;
goto <bb3>;
goto <bb5>;
<bb3>:  i = i + 1;
<bb4>:
  if (i <= 9) goto <bb3>;
  else       goto <bb7> ;
<bb5>:   j = j + 1;
<bb6>:
  if (j <= 9)  goto <bb5>;
  else         goto <bb7>;
<bb7>:  k = i + j;
```

(b)

Figure 1: A Simple C-like program

Algorithm 1 STRUCT2TUPLE **subNetForAssignMentBB** (b, N, PB)

Require: A basic block, a bipartite graph N, set of parallel blocks
Ensure: Two tuple structures. The elements of this structure are as follows: 1.updated bipartite graph and 2. parallel block list

```
$G = \emptyset$;
$G =$ ———-(a) $\cup$ $creatDDG$ ($b$);
/*Construction is carried out by GauTe Tool (or use fdump statement) */
$L = reachingDefinitionAnalysis$ (————(b));
/* The function returns a set of lists. Each list contains set of statements. Every statement in a list is independent to the other statements present in that list.
for each list $l$ in $L$ do
    $P =$ ——————(c);
    for each element $e$ in $l$ do
        $P = P\cup$————-(d); /* The function takes an element and creates places $p$ for every used variable of that element */
        $T = T\cup$————-(e);
        for each $t$ in $T$ do
            /* Construct normalize expression and guard condition (as per taught in the class)*/
        end for
        $P_{out} = P_{out}\cup$———(f);
        /*The function creates an output place for the transition $T$ and update the symbol table for places and transitions */
        Attach $p, t$ and $p_{out}$
    end for
end for
if number of block associated with goto ——(g)(conditional operator) 1 then
    The blocks along with goto statement are put in to $PB_{new}$;
    $PB = PB\cup$—————(h); //update the parallel block lists
end if
Update $N$
return $\langle - - - - -(i), - - - - - - - - - -(j)\rangle$;
```

END