Roll No.: _ _ _ _ _ _ _ _ _ _ _ _ _ _ _

Amrita Vishwa Vidyapeetham

Amrita School of Engineering, Coimbatore

B.Tech. Degree Examinations – April/May 2018

Sixth Semester

Computer Science and Engineering

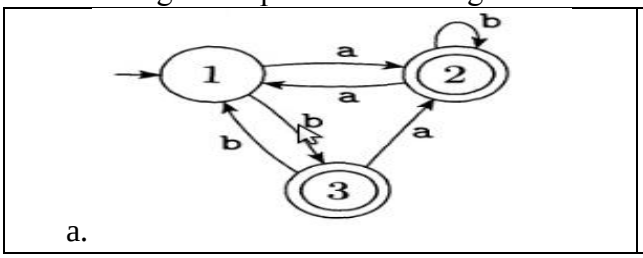
15CSE311 Compiler Design

[Time: Three hours Maximum : 100 Marks]

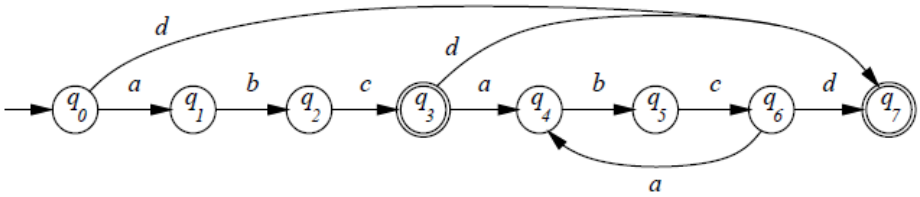
Answer all questions

PART A **(10 x 4 =40 Marks)**

1. Find the regular expression for the given automata.



2. For the given finite automaton and the input string $(abc)^4$, find the output obtained by Maximal Munch Scanner.



3. Check if the given grammar is ambiguous. If yes, give reason and remove ambiguity. If not, give reason.

Expr→Expr + **Expr**
| **Expr Expr**
| **Expr***
| id

4. Show the behaviour of LR(1) parser on the input string *id := id + id* for the given grammar based on the table given as stack, input string, handle and action format. For every step of the stack show the individual parse tree growth.

[1] S' → S
[2] S →S ; A
[3] S → A
[4] A → E
[5] A →id := E
[6] E → E + id
[7] E → id

R

State	id	;	+	:=	$	S	A	E
0	s4					1	2	3
1		s5			accept			
2		r3			r3			
3		r4	s6		r4			
4		r7	r7	s7	r7			
5	s4						8	3
6	s9							
7	s11							10
8		r2			r2			
9		r6	r6		r6			
10		r5	s6		r5			
11		r7	r7		r7			

5.

a. A shift-reduce parser carries out the actions specified within braces immediately after reducing with the corresponding rule of the grammar given below. Using the syntax directed translation scheme described by the following rules specify the translation of *xxxxyzz*.

S → xx W {print"1"}
S → y {print"2"}
W → Sz {print"3"}

b. Consider the following Syntax Directed Translation Scheme (SDTS), with non-terminals {S, A} and terminals {**a, b**}.

$S \rightarrow aA$ { *print* 1 }
$S \rightarrow a$ { *print* 2 }
$A \rightarrow Sb$ { *print* 3 }

Processing the above SDTs, What is the output printed by a bottom-up parser, for the input **aab**?

6. For the sequence of instructions shown below depict an SSA-form representation (as there could be more than one). Do not forget to include the φ-functions.

```
a = b * 3.0;
if (a < 0) {
        a = 0;
        b = b + 1;
} else
{
        b = 0;
}
z = a;
y = b;
```


R

7. Construct a Control Flow Graph for the following ‘C’ Program:

```
int main()  {
        int x  =  5;
        while (x  >  0)  {
        printf("Hello!");
        --x;
        }
      return 0;
    }
```

8. What is the “output” of the below pseudo code assuming bindings are resolved using the closest nested scope rule in the following conditions:
    a. dynamic scoping
    b. lexical (i.e., static) scoping?

```
VAR count: INTEGER;

PROCEDURE procX IS
BEGIN
        VAR count: INTEGER;
         SET count TO 100;
        CALL report
END

PROCEDURE procY IS
BEGIN
        SET count TO 200;
        CALL report
END

PROCEDURE report IS
BEGIN
        PRINT “count = ” + count
END

MAIN PROGRAM IS
BEGIN
        SET count TO 300;
        CALL procX;
        CALL procY
END
```


R

9. For each of the following actions, state whether it happens at compiler construction time (i.e., when developing the compiler), at compile time (i.e., when compiling or linking a program), or at run-time (i.e., when executing the code generated by the compiler).
    a. Compute the numerical offset for a local variable within a stack frame for the C programming language.
    b. Choose the order of evaluation of the operators in a single expression.
    c. Choose the representation (data structures) to use for a 3-address code intermediate representation.
    d. Choose whether to use the mid-level or low-level model of compilation.
10. Give the general structure of an activation record for a procedure. Discuss the significance of the fields of a record.

PART B **(5 x 12 =60 Marks)**

11. Consider the following grammar fragment for an expression.

    $exp \rightarrow \text{CONST} \mid \text{IDENT}_1 \mid \text{IDENT}_2[exp_1]$

    Assume that the nonterminal *exp* and terminals IDENT and CONST have an attribute type that can be assigned a range of type values, including *typeBoolean*, *typeInt*, *typeError*, etc. In addition, the terminal IDENT has an attribute *isArray* that can be true or false.
    a. Add syntax-directed type checking rules to the production exp → $\text{IDENT}_2[exp_1]$ to enforce the following: (6 Marks)
        - IDENT_2 must be an array. Otherwise call parser.msg("Misuse of array").
        - The subscript expression exp1 must be *typeInt*. Otherwise call parser.msg("Integer required").
        - The expression type is assigned the type of IDENT_2 if both (a) and (b) are true, else it is assigned *typeError*.

    b. Give the parse tree for x[y[2]], where x is a BOOL array, y is an INT array, and 2 is an INT, show how syntax directed translation can calculate the type of the expression using your type checking rules by annotating the parse tree. Show the value calculated for each attribute, and draw arrows linking the attributes used in the calculation. (6 Marks)

12.
    a. Structural IR is considered high level and Linear IR is considered level. Justify with an example. (5 Marks)
    b. For the sequence of instructions shown below depict an SSA-form representation (as there could be more than one). Do not forget to include the f- functions. Also discuss what sort of optimizations could be exploited in this specific case, meaning, using the information about which definitions reach which uses and derive specific variable values. (8 Marks)

R

```
        b = 0;
        d = 0;
        a = 1;
        i = ...;
Lloop:  if(i > 0){
           if(a < 0){
              b = i;
            } else {
              b = 0;
            }
            i = i - 1;
            if(i < 0) {
             i = 0;
             goto Lbreak;
            }
            if (b == 0){
             goto Lbreak;
            } else {
             a = a + d;
            }
            goto Lloop;
          }
Lbreak:   x = a;
          y = b;
```

13. Consider the grammar:

    S → if E then S1

    S → if E then S1 else S2

    S → while E do S1

a. Provide the semantic rules to generate the intermediate code as discussed in class.

b. Show the generated code for the snippet of code below using the short-circuit scheme for the code generation of predicates. State the assumptions you make and use symbolic labels for targets of jump and draw a memory layout of the code.

```
i = 0;
while ((a < b) and (i < 10))
{
        if (c[i] < 0)
        {
                break;
        }
        i = i + 1;
}
```

c. Suggest an improvement over part (b) that will result in fewer instructions being executed.

R

14. For the PASCAL code below answer the following questions:

```
01:   procedure main
02:     integer a, b, c;
03:     procedure f1(a,b);
04:       integer a, b;
05:       call f2(b,a);
06:       end;
07:     procedure f2(y,z);
08:      integer y, z;
09:      procedure f3(m,n);
10:       integer m, n;
11:       end;
12:      procedure f4(m,n);
13:       integer m, n;
14:       end;
15:      call f3(c,z);
16:      call f4(c,z);
17:      end;
18:   ...
19:   call f1(a,b);
20:   end;
```

a. Draw the symbol tables for each of the procedures in this code (including main) and show their nesting relationship by linking them via a pointer reference in the structure (or record) used to implement them in memory. Include the entries or fields for the local variables, arguments and any other information you find relevant for the purposes of code generation, such as its type and location at run-time. (9 Marks)

b. For the statement in line 15, what are the specific instances of the variables to be locate ? Explain how the compiler obtains the data corresponding to each of these variables table. (3 Marks)

15. Consider the following PASCAL source program shown below.

```
01: program main(input,  output);
02:     var p[1..2]: integer;
03:     var z: integer;
04:    procedure P1(a: integer);
05:      begin
06:        z := a+1;
07:        writeln(p[z])
08:      end;
09:     procedure P2(b: integer);
10:      begin (* P2 *)
11:        z := 0;
12:        P3(b)
13:      end;
14:    procedure P3(c: integer);
15:      begin (* P3 *)
16:        p[c] := z;
17:        P1(0);
18:      end;
19:  begin (* main *)
20:    P2(1)
21:  end.
```

a. Discuss if the Activation Records (ARs) for each of the procedures P1 through P3 can be allocated statically or not. Explain why or why not. (2 Marks)

b. Assuming that stack is being used, draw its layout. (8 Marks)

c. For this particular code do you need to rely on the Access Links on the AR to access nonlocal variables? Would there be a substantial advantage to the use of the Display mechanism? (2 Marks)

****

R