A **Reg. No. :** | | | | | | | | | | | | |

Question Paper Code: U5C02

B.E. / B.Tech. DEGREE EXAMINATION, APRIL 2024

Fifth Semester

Computer Science and Business Systems

21UCB502- COMPILER DESIGN

(Regulations 2021)

Duration: Three hours Maximum: 100 Marks

Answer ALL Questions

PART A - (10 x 1 = 10 Marks)

1. Compiler should report the presence of _____ in the source program, in translation process. CO1-U

   (a) Classes (b) Objects (c) Errors (d) Text

2. Tokens are specified by CO1-U

   (a) Regular expressions (b) Algebraic expressions

   (c) Arithmetic expressions (d) Boolean expressions

3. A grammar that produces more than one parse tree for some sentence is called as ----- CO1-U

   (a) Ambiguous (b) Unambiguous (c) Regular. (d) none

4. The grammar s→ L=R leads to the presentation of CO1-U

   (a) Arithmetic expression (b) Left to right expression

   (c) Shift from left to right expression (d) An assignment statement

5. The postfix equivalent of **(a+b)*(c+d)**is ---------- CO2-App

   (a) ab+cd+* (b) ab+cd*+ (c) *+ab+ab (d) ab+*ab+

6. Syntax Directed Translation is ---- CO1-U

   (a) Production with Semantic actions (b) Production with LR(0) items

   (c) Production with LR(1) items (d) Production only

7. In Algebraic expression simplification, a = a + 1 can simply be replaced by? CO1-U

(a) a (b) INC a (c) DEC a (d) MUL a

8. Code generator uses _____ function to determine the status of available registers and the location of name values. CO1-U

(a) setReg (b)cinReg (c) pfReg (d) getReg

9. The technique of replacing run time computations during compile time is called------ CO1-U

(a) Constant folding (b) code hosting

(c) peephole optimization (d) invariant computation

10. What is the equivalent optimized expression of x=x*2? CO1-U

(a) x=x/2 (b) x=x*0.5 (c). x=x+x (d) x=x-x

PART – B (5 x 2= 10 Marks)

11. Draw the transition diagram for the given Regular Expression (0/1) (0/2). CO1-U

12. Find FIRST and FOLLOW for the given grammar CO2-App

S→CC

C→cC

C→d

13. Translate a given statement into three address code CO2-App

While (I<10){

I=I+1; }

14. What are the different types of issues in the design of code generator? CO1-U

15. Optimize the following loop CO2-App

```
sum=0;
i=1; a,b=5;c=8;
while(i<=10) do
{
    a=b+c;
    sum=sum+i*I;
    i++;
    }
```

PART – C (5 x 16= 80 Marks)

16. (a) Illustrate how the following high level language statement is transformed into machine code during the compilation process **x=(a+b) * (c+d)** with the neat sketch of phases of compiler. CO2-App (16)

Or

(b) Derive DFA for the given regular expression (a|b)*abb. CO2-App (16)

17. (a) Design non-recursive predictive parser for the following grammar and parse the string **id+id*id**. CO2-App (16)

E→E+T |T
T→T*F |F
F →(E) |id

Or

(b) Design a predictive parser for the following grammar and also and parse the string **(a)** CO2-App (16)

S→ a | ↑| (T)
T →T, S | S

18. (a) (i) Translate the statement CO2-App (8)

```
c=0
do{
   if (a>b) then
         x++
   else
         x - -
         c + +
} while(c<5)
```

(ii) Translate the statement CO2-App (8)

```
switch (ch)
{
Case 1:
     C=a+b;
     Break;
Case 2:
     C=a-b;
     Break;
}
```

Or

(b) (i) Draw the quadruple, triple and indirect triples for the foll statement a = (b*c) + (a+b+c). CO2-App (8)

(ii) Draw the quadruple, triple and indirect triples for the foll statement (a + b) * (c + d) - (a + b + c) CO2-App (8)

19. (a) For the statement d:=(a-b)+(a-c)+ (a-c) generate three address code and subsequent target code using the simple code generation algorithm CO2-App (16)

Or

(b) Develop a quicksort algorithm for reads nine integers into an array and sorts them by using the concepts of activation tree. CO2-App (16)

20. (a) Construct the DAG for the following basic block CO2-App (16)

1. t1:=4*i
2. t2:=a[t1]
3. t3:=4*i
4. t4:=b[t3]
5. t5:=t2*t4
6. t6:=prod+t5
7. prod:=t6
8. t7:=i+1
9. i:=t7
10. if i<=20 goto 1

Or

(b) Convert the following three address code into flow graph and optimize the flow graph CO2-App (16)

1. i=1
2. j=1
3. t1=10*i
4. t2=t1+j
5. t3=8*t2
6. t4=t3-88
7. a[t4]=0
8. j=j+1
9. if j<=10goto 3
10. i=i+1
11. if i<=10 goto 2
12. i=1
13. t5=i-1
14. t6=88*t5
15. a[t6]=1
16. i=i+1
17. if i<=10 goto 13