B.E./B.TECH. Degree Examination, December 2020

Semester - VI

IT16604 - Automata and Compiler Design

(Regulation 2016)

Time: Three hours Maximum : 80 Marks

Answer **ALL** questions

PART A - (8 X 2 = 16 marks)

1. Determine the number of states needed to accept a string ends with 10.
   a) 2
   b) 3
   c) 1
   d) can't be represented
2. The grammar S → aSa | bS | c is
   a)LL(1) but not LR(1)
   b)LR(1)but not LR(1)
   c)Both LL(1)and LR(1)
   d)Neither LL(1)nor LR(1)
3. Which one of the following is FALSE?
   a) A basic block is a sequence of instructions where control enters the sequence at the beginning
   and exits at the end.
   b) Available expression analysis can be used for common subexpression elimination.
   c) Live variable analysis can be used for dead code elimination.
   d) x = 4 ∗ 5 => x = 20 is an example of common subexpression elimination.
4. In analyzing the compilation of a program, the term "Machine independent optimization" is assosiated with
   a) recognization of basic syntactic construction through reductions
   b) recognition of basic elements and creation of uniform symbols
   c) creation of more optical matrix
   d) use of macro-processor to produce more optimal assembly code
5. Differentiate NFA and DFA
6. Eliminate left recursion in the grammar A → Ac | Aad | bd
7. How is liveliness of a variable calculated?
8. How to optimize an expression using peephole optimizations?

PART B - (4 X16 = 64 marks)

9. (a) (i) Construct DFA with its transition table for the following languages **(8)**

   i. The set of strings over {a,b,c} having bca as substring.

   ii. L={ $a^n b^m c^p$ | n,m,p ≥ 1 }

   (ii) Construct NFA with its equivalent DFA for the regular expression (a+b)*ab(a+b)* **(8)**

(OR)

   (b) Convert the regular expression (a+b)*ab to DFA using Subset Construction method and determine the minimized DFA. **(16)**

10. (a) Describe the various phases of compiler and apply the analysis phase of compiler for an expression a:=b+c*5 **(16)**

(OR)

   (b) (i) Construct stack implementation of shift reduce parser for the grammar **(8)**

   E → E + E

   E → E * E

   E → (E)

   E → id

   and show whether the input string id1 + id2 *id3 will be accepted or not.

   (ii) Determine the parsing table using LL(1) parser for the grammar **(8)**

   S->A a A b | B b B a

   A-> ε

   B-> ε

11. (a) (i) For the productions given below, write the semantic rules and draw the annotated parse tree for the expression (9+8*(7+6)+5)*4 **(8)**

   L-> E

   E-> E+T | T

   T-> T*F | F

   F-> (E) | digit.

   (ii) Generate intermediate code for the expression (a*b) +(c-d) *(a*b) +b **(8)**

(OR)

   (b) (i) Show how backpatching helps to generate the address with a suitable example. **(8)**

(ii) Illustrate type checking with necessary diagram **(8)**

12. (a) (i) Construct DAG for the expression x=((a+b)/(b-c))-(a+b)*(b-c)+f **(8)**

(ii) What is run time stack? Illustrate storage allocation strategies used for recursive procedure calls. **(8)**

(OR)

(b) Generate Intermediate code for the following code segment and optimize the code by applying principle sources of optimization **(16)**

```
void insertionSort(int arr[], int n)
{
int i, key, j;
for (i = 1; i < n; i++)
{
key = arr[i];
j = i - 1;
while (j >= 0 && arr[j] > key)
{
arr[j + 1] = arr[j];
j = j - 1;
}
arr[j + 1] = key;
}
}
```