National Institute of Technology, Tiruchirapppalli - 15
Department of Computer Science and Engineering
End Semester Examination – July 2021
Set - 4
CSPC41 – Principles of Compiler Design

Course/Department **: B.Tech./CSE** Batch : **2018-2022**
Semester/Section **: VII** Session **: JUL/2021**
Date and Time **: 04-12-2021 & 12.00 PM – 02.00 PM** Marks **: 30**

Answer ALL Questions with proper steps and justification.
Include diagrams or tables wherever necessary.

1. Consider the statement x = b + a ^ c / 4.5 - 3. Discuss on the various phases of the compiler by showing changes on this statement after each phase where variables a, b, and c are of integer data type and x is real. **(3)**

2. Write a program in LEX to count the occurrence of vowels in a string. **(2)**
Sample Input: I love my country
Output: 5

3. Compute FIRST and FOLLOW for the grammar given below. **(2)**
$S \rightarrow BCD|aD$
$A \rightarrow CEB|\ aA$
$B \rightarrow b|\epsilon$
$C \rightarrow dB|\ \epsilon$
$D \rightarrow cA|\epsilon$
$E \rightarrow e|fE$

4. Construct LALR parsing table for the following grammar. Parse a string of minimum length 7 using the parsing table constructed by handling S-R conflicts if present. **(4)**
$S \rightarrow L := E$
$E \rightarrow E + E \mid (E) \mid L$
$L \rightarrow Elist] \mid id$
$Elist \rightarrow Elist, E \mid id\ [\ E$

5. Write three-address code for the following code fragment. Identify the basic blocks in the resultant three-address code and construct control flow graph. Perform code optimization in three-address code wherever possible. **(5)**

```
int low=-2, high=100, i, flag;
  while (low < high) {
    flag = 0;
    if (low <= 1) {
      ++low;
      continue;
    }
    for (i = 2; i <= low / 2; ++i) {
      if (low % i == 0) {
        flag = 1;
        break;
      }
    }
    if (flag == 0)
      printf("%d ", low);
    ++low;
  }
```

6. Perform code generation for the expression (a*b/-t)+c^(d+e)*x using dynamic programming. Assume that, two registers are available which can be used for computation. Show the syntax tree with cost vector at each node. **(3)**

7. Construct DAG for the following basic block. **(2)**

```
t1 := b + c
t2 := c / d
t3 := b + c
b := c / d
d := b + c
```

8. Give the three-address code in quad-triples representation for the expression **(2)**
z := a[m]^(b*-m+1)/c*(d%e)-e.

9. Write target code in assembly language (hypothetical machine: cost of all operations are the same and extra cost for operations involving memory) for the following three-address code fragment and find its cost. You can assume that variable *i* and *x* is already loaded to register R0 and R1 respectively by some statements in the code before line 101, whereas registers for remaining variables are not allocated yet. In order to perform an arithmetic operation, at least one of the operands should be in register. **(4)**

101: *y* := *b[i]*
102: *y* := *y / x*
103: *x* := *a * k*
104: **v* := *x - y*

10. Compute reaching definitions on the following code using data flow equations and show the annotated syntax tree with reaching definitions. **(3)**

```
j = 1;
a = 2;
i = m;
k = 2;
while i < j and j < n or k < n do
        c = 1;
        if c < j then
                x = x + c;
        else
                if c > a then
                        x = 2 * c;
                        a = a + 1;
                else
                        x = c * a;
                        a = a - 1;
        c = c + 1;
        j = j + 1;
        k = k * 2;
end
a = x + n;
```