School of Engineering and Applied Sciences
Department of Computer Science and Engineering

Compiler Design (CSE 306)

TUTORIAL – 7

Submission Instructions:

Submission should be handwritten.

- Write your answers on A4 sheet, scan and submit.
- You can also submit a digital version like OneNote or any other notes taking apps, but it should be handwritten.
- Handwriting will be compared and punished if found plagiarized.

Problem Set:

SET -1

1. For the SDD of variable declaration grammar discussed in class, give annotated parse tree and dependency graph for the following statement.
   float x,y,z,w

2. An SDD is S-attributed if every attribute is synthesized and L-attributed if each attribute must be either synthesized or inherited (with information flow only from its left siblings). Consider the production A→BCD. Each of the four non-terminals A, B, C, and D have two attributes: s: a synthesized attribute, and i: an inherited attribute. For each of the sets of rules given below, tell whether (i) the rules are consistent with an S-attributed definition (ii) the rules are consistent with an L-attributed definition.
   a. A.s = B.i + C.s.
   b. A.s = B.i + C.s and D.i = A.i + B.s.
   c. A.s = B.s + D.s.
   d. A.s = D.i, B.i = A.s + C.s, C.i = B.s, and D.i = B.i + C.i.

3. Construct a Syntax-Directed Translation scheme that translates arithmetic expressions from infix into postfix notation. Your solution should include
   a. The context-free grammar
   b. The semantic attributes for each of the grammar symbols
   c. Semantic rules.

   Show the application of your scheme to the input and "3*4+5*2".

4. Consider the following grammar G for expressions and lists of statements (StatList) using assignment statements (Assign) and basic expressions (Expr) using the productions presented below.

    StatList →Stat ; StatList
    StatList →Stat
    Stat →Assign
    Expr →Expr + Expr
    Expr →int
    Expr →id
    Assign →Expr = Expr
    Assign →Expr += Expr

   Using a stack-based machine write a syntax-directed definition to generate code for StatList, Assign and Expr. In this stack-based machine you need to push the operands of the expressions first onto the stack and them execute the operations with the topmost element(s) of the stack. For example, the input "a = b + 5" where the identifiers "a" and "b" have already been defined previously, would result in the following generated code:

    push b
    push 5
    add
    top a

    Here the instructions "push" simply puts the value of their argument on top of the stack. The instruction "add" adds the two topmost elements of the stack and replaces them with the numeric value corresponding to the addition of the numeric values at the top of the stack before the instruction executes. The instruction "top a" copies the top of the stack and assigns the value to the variable "a". The value of "a" remains at the top of the stack as it can be used as the argument for another operand.

5. Consider a grammar for signed binary numbers

    Number → sign list
    sign → + | -
    list → list bit | bit
    bit → 0 | 1

    a. Build attribute grammar that annotates Number with the decimal value it represents
    b. Draw the dependency graph for the binary number: -100

6. Write type expression for the identifiers tensor, cgpa and student in the following declarations.
    a. int tensor[5][6][7].
    b. int cgpa(char id[10],int *sem_gpa)
    c. typedef struct stud{
       int roll_no,
       char *name,*address,
       int sem_gpa[8]
       }student