# UNIT – III

1. Construct an **annotated parse tree** for a given expression.
2. Construct a **syntax tree** for an arithmetic expression.
3. Define **Syntax-Directed Definition (SDD)**.
4. Explain **simple type declarations** for `int` and `float`.
5. Explain the working of a **simple desk calculator** using syntax-directed translation.
6. Convert an **infix expression to postfix notation**.
7. Convert an **infix expression to prefix notation**.

---

# UNIT – IV

1. Explain the **different variants of syntax trees**.
2. Explain **type declaration** in programming languages.
3. Explain **type checking** and its importance in a compiler.
4. Write an **SDD for computing the type and width of basic types**.
5. Write an **SDD for computing the type and width of arrays**.
6. Explain **Quadruples** with example.
7. Explain **Triples** with example.
8. Explain **Indirect Triples** with example.
9. Explain **address translation for Boolean expressions**.
10. Explain **address translation for control statements**.
11. Explain **address translation for switch statements**.
12. Explain the **backpatching scheme** with suitable examples.

---

# UNIT – V

1. Explain the **code generation algorithm**.
   Generate target code for a given arithmetic expression.
2. Explain **code generation for simplified procedure calls and returns**.
3. Explain **Common Sub-Expression Elimination (CSE)** with a suitable example.
4. Explain the **main issues involved in code generation**.

5. What is **Three-Address Code (TAC)**?
   Construct the **basic blocks** for the following program segment:

```
rev = 0;
while (num >= 0) {
    dig = num % 10;
    rev = rev * 10 + dig;
}
```

6. Explain the **construction of a Directed Acyclic Graph (DAG)** for a basic block.
7. Explain the **different types of instructions used in a simple target machine model**. Briefly explain each type.
8. Explain **Common Sub-Expression Elimination and Dead Code Elimination**.