Q1. The compiler compiles following expression; Write phase-wise results in the compilation process.

R = ( b \* b – 4 \* a ) / (2 \* a)

Q2. Explain the LEX file structure with a suitable example along with the execution steps.

Q3. What is machine dependent optimization and machine independent optimization techniques? Discuss dead code elimination and constant propagation techniques.

Q4. Write the following expression in the form of postfix notation, DAG, 3 address code with quadruple and triples.

P = (x + y) \* (y + z) + (x + y + z)

Q5. Write Three Address Code for the following expression-

If A < B and C < D then t = 1 else t = 0

Q6. Elaborate intermediate code generation, memory management, and register allocation issues involved in this conversion?

Q7. What is static and dynamic type checking? List various type checking rules for expressions.

Q8. What is the role of code optimization in the compilation process? Explain any three function preserving transformation techniques.

Q9. The compiler compiles the following expression; Write phase-wise results in the compilation process: a, b, c are of float type.

z = a + b \* c \* 2

Q10. Consider the grammar -

E → E – E

E → E \* E

E → id

How top-down parser helps in deriving the input “5 – 3 \* 4”? Show appropriate steps involved.

Q11. How top-down parsing differs from bottom-up parsing method?

Q12. Write the following expression in the form of postfix notation, DAG, 3 address code with quadruple and triples.

X = ( b \* -c ) + y + (b \* -c) / z

Q13. Write Three Address Code for the following expression-

if (a < b + c)

a = a - c;

c = b \* c;

Q14. Elaborate target program, instruction selection, and evaluation order issues involved in this conversion?

Q15. What is static and dynamic type checking? List various type conversion rules for expressions.