Certainly! Here's a Compiler Design practical exam paper with 3 questions, each worth 30-40 marks:

Question 1: Lexical Analysis (40 Marks)

Write a Lex program to recognize tokens for a simple programming language that includes the following components:

1. Keywords: **if**, **else**, **while**, **int**, **float**
2. Identifiers: Any sequence of letters and digits starting with a letter.
3. Constants: Integer and floating-point constants.
4. Operators: **+**, **-**, *****, **/**, **=**, **==**, **!=**, **<**, **>**, **<=**, **>=**
5. Delimiters: **(**, **)**, **{**, **}**, **;**

Ensure that your Lex code recognizes each token type and prints out the corresponding token along with its lexeme.

Question 2: Syntax Analysis (30 Marks)

Given the following context-free grammar for a simple language:

phpCopy code

<stmt> ::= <if_stmt> | <assignment> <if_stmt> ::= if (<expr>) <stmt> <assignment> ::= <id> = <expr> <expr> ::= <id> | <constant> | <expr> <op> <expr> <id> ::= (any valid identifier) <constant> ::= (integer or floating-point constant) <op> ::= + | - | * | / | == | != | < | > | <= | >=

Write a Lex program to tokenize the input source code and a Yacc (or Bison) program to perform syntax analysis according to the given grammar. Ensure that your Yacc program can recognize valid statements and expressions and can handle nested if statements.

Question 3: Semantic Analysis and Intermediate Code Generation (30 Marks)

Extend the previous Lex and Yacc programs to perform semantic analysis and generate intermediate code for the given source code. Implement semantic rules to check for the following:

1. Variable declarations and assignments should match types.
2. Ensure that only boolean expressions are used in the if statement conditions.
3. Generate intermediate code (e.g., three-address code) for assignment statements and if statements.

Provide sample input source code and the corresponding intermediate code generated by your program for testing