CS348 Lab Test: Calculator Implementation

Date: 25th March, 2025

You are required to implement a calculator that supports the following operations:

1. Expression Evaluation
    a. Syntax: expr(expression)
    b. Evaluates a mathematical expression containing addition (+), subtraction (-), multiplication (*), division (/), and exponentiation (**).
    c. Example: expr(5+4*5/2-4**2) => -1

2. Binary to Decimal Conversion
    a. Syntax: bintodec(binary_string)
    b. Converts a binary number (as a string) to its decimal equivalent.
    c. Example: bintodec(0001101010) => 106

3. Binary to Hexadecimal Conversion
    a. Syntax: bintohex(binary_string)
    b. Converts a binary number (as a string) to its hexadecimal equivalent.
    c. Example: bintohex(010101000) => A8

4. Hexadecimal to Binary Conversion
    a. Syntax: hextobin(hex_string)
    b. Converts a hexadecimal number (as a string) to its binary equivalent.
    c. Example: hextobin(2F) => 00101111

5. Hexadecimal to Decimal Conversion
    a. Syntax: hextodec(hex_string)
    b. Converts a hexadecimal number (as a string) to its decimal equivalent.
    c. Example: hextodec(1A3) => 419

Implementation Requirements:

(name of all files must be your roll number)

1. Lex Code:
    a. Tokenize input expressions.
    b. Identify keywords (expr, bintodec, bintohex, hextobin, hextodec).
    c. Recognize numbers & operators (decimal, binary, hex, +, -, *, /, **).
    d. Pass tokens to Bison for parsing.

2. Bison Code:
    a. Implement grammar rules.
    b. Perform appropriate reductions.
    c. Evaluate expressions and conversions.

3. Makefile:
    a. Create a Makefile for compilation and execution.
    b. The default rule should compile Lex and Bison files to generate an executable.

Sample Test Cases:

Input	Expected Output
expr(3+5*2-4**2/2)	5
bintodec(1101)	13
bintodec(00001111)	15
bintohex(10101001)	A9
bintohex(11110000)	F0
hextobin(1F)	00011111
hextobin(A3)	10100011
hextodec(FF)	255
hextodec(2A3)	675