

Programming Assignment #1 – Due Monday, 8 October

Problem Statement:

For this programming assignment, you will gain some experience programming with Flex, Bison and C. You will modify the Flex and Bison specifications given in class that implement a simple integer expression calculator by adding new features including registers, arithmetic and relational operators, a conditional and simple functions. You will have to add new lexical specifications rules and BNF grammar rules for correctly evaluating the new types of expressions. The operators and their associativity rules are given below, listed from lowest to highest precedence. Parenthesis “()” are used to group subexpression to override default precedence rules. As a starting point you should use the iCalc 1.0 files: icalc.l, icalc.y handed out in class.

Operators	Category	Associativity
=, +=, -=, *=, /=, %=	binary assignment	right-to-left
?:	ternary conditional	right-to-left
	binary logical or	left-to-right
&&	binary logical and	left-to-right
<, >, <=, >=	binary relational	left-to-right
==, !=	binary equality/inequality	left-to-right
+, -	binary additive	left-to-right
*, /, %	binary multiplicative	left-to-right
^	binary exponentiation	right-to-left
-, +	unary minus/plus	right-to-left
++, --	postfix incr/decr	left-to-right
++/--	prefix incr/decr	right-to-left

iCalc 2.0 – a basic integer calculator

For this assignment you are to extend icalc 1.0 to icalc 2.0 with the following new features.

1. Add twenty-six 32-bit integer registers, labeled a-z, so that users can assign values to registers and use those registers in subsequent expressions in the calculator. A register variable by itself evaluates to the value stored in the register. Registers also require that you implement assignment expressions of the form `reg = expr`, `reg += expr`, `reg -= expr`, `reg *= expr`, `reg /= expr`, and `reg %= expr`. Each assignment expression evaluates to the value of the expression on the RHS, and updates the register on the LHS, printing the result. Registers should all be initialized to 0 when the calculator starts. Note that all assignment operators are *right associative*, so assignment expressions of the form: `reg = reg = ... = reg = expr` are legal. For example:

```
icalc> a
0
icalc> a=5
5
icalc> a *= 3 + b
15
icalc> a = b = c = 9
9
icalc> z += a * b + c
90
```

2. Add the C language unary pre/post-increment (++) and pre/post-decrement (--) operators ensuring that they can only be applied to *registers* (i.e., ++2 is a syntax error). Note that these unary operators have higher precedence than binary operators. The result of applying these operators should have the same semantics as in C. For example:

```
icalc> x=1
1
icalc> x++
1
icalc> x
2
icalc> ++x
3
icalc> ---x
syntax error
icalc> -(--x)
-2
```

3. Implement built-in constants MININT and MAXINT assuming signed 2's complement 32-bit integers.

```
icalc> MAXINT
2147483647
icalc> MININT
-2147483648
```

4. Implement a binary *right associative* exponentiation operator (^) that computes the first argument raised to the power of the second argument. Note that a standard C function pow(x, y) that computes x^y is defined in math.h. The pow function computes x^y up to MAXINT and $-x^y$ up to MININT, and $\text{pow}(x, 0) = 1$ for all x. For example:

```
icalc> 2^0
1
icalc> 2^10
1024
icalc> -2^31
-2147483648
icalc> 2^31
2147483647
icalc> 2^100
2147483647
```

5. Implement built-in functions abs, min and max as prefix operators that operate on a comma separated parenthesized list of arguments that can be constants, registers or (eagerly evaluated) expressions: For example:

```
icalc> abs(-11)
11
icalc> x = abs(a - 2 * 3)
6
icalc> min(abs(-2), min(5, 1))
1
icalc> max(9, 2*5)
10
```

6. Implement relational operators `==`, `!=`, `<`, `<=`, `>`, `>=`, logical operators `||` and `&&`, and a *ternary* conditional expression operator `? :`. Note that a relational expression evaluates to the constant “true” (1) or “false” (0). Ternary conditionals may be nested. For example:

```
icalc> 4 == 2 * 2
1
icalc> (a >= MININT && a <= MAXINT) ? 1 : 0
1
```

Note that the syntax of the conditional expression is: *(bool expr) ? int expr : int expr*. Where the Boolean expression evaluates to either true or false.

7. Implement a procedure `swap(a,b)` that swaps the contents of any two registers, but does not print out any result.

Submission Requirements:

You are to hand in at the start of class on the due date source code listings of all of your program files stapled together. Your final source code listings must be printed using the `enscript` command: `enscript -C -2Gr -Ec files...` Enscript will print your files 2-up rotated landscape mode (-2Gr) with line numbers (-C) and with C syntax highlighting (-Ec). You must also to submit an electronic copy of your source program using the **turnin** program on CS Department Linux machines BEFORE class begins on the due date. See the course newsgroup for specific instructions. **The TA will post to the class newsgroup the exact instructions for turning in the assignment using the turnin program. No late hardcopy or electronic programs will be accepted.**

Honor Policy:

This assignment is an individual learning opportunity that will be evaluated based on your ability to think independently, work through a problem in a logical manner and implement a software program on your own. You may however discuss verbally or via email the general nature of the conceptual problem to be solved with your classmates, the course TA or the course instructor, but you are to complete the actual programming for this assignment without resorting to help from any other person or other resources that are not authorized as part of this course. If in doubt, ask the course instructor. *Any indication of improper cooperation or the use of non-approved course materials will result in a grade of zero being assigned for this project and referral to the College for possible disciplinary action with respect to your continued participation in this course. **By submitting a solution to this assignment under your name, you are asserting that the solution was done by you under this honor policy.***