Alexander J King

Comparative Programming Languages

Homework #7

1) Create a program in C++ or C# that forces one data type to be converted into another (using explicit type conversion). Show the Source code.

int a = 1;

Console.WriteLine(a);

float a;

Console.WriteLine(a);

2) What is short-circuit evaluation?

• An expression in which the result is

determined without evaluating all of the

operands and/or operators

• Example: (13 \* a) \* (b / 13 – 1)

If a is zero, there is no need to evaluate (b /13

- 1)

3) Define functional side effect.

Functional side effects: when a function changes a

two-way parameter or a non-local variable

4) Define operator precedence and operator associativity.

The operator associativity rules for expression

evaluation define the order in which adjacent

operators with the same precedence level are

evaluated

• Typical associativity rules

– Left to right, except \*\*, which is right to left

– Sometimes unary operators associate right to left (e.g., in

FORTRAN)

• APL is different; all operators have equal

precedence and all operators associate right to left

• Precedence and associativity rules can be overriden

with parentheses

5) Do you think the elimination of overloaded operators in a language would be beneficial? Why or why not?

No. Function overloading solves the problem of sorting for datatypes.

6) Assume the following rules of associativity and precedence for expressions:

Precedence:

Highest \*, /, not+, –, &, mod– (unary)=, /=, < , <=, >=, >and

Lowest or, xor

Associativity Left to right

Show the order of evaluation of the following expressions by parenthesizing all subexpressions and placing a superscript on the right parenthesis to indicate order.

For example, for the expression a + b \* c + d the order of evaluation would be represented as ((a + (b \* c)1)2 + d)3

a. (a \* b) – (1 + c)

b. {a \* (b – 1) }/ (c mod d )

c. {(a – b) / c} & (d \* e / a – 3)

d. -a or c = d and e – no edit?

e. (a > b) xor (c or d )<= 17

f. -a + b

Now show the order evaluation of the expression (in a thru f), assuming that there are no precedence rules and all operators associate right to left.

a\*b-1+c

a\*b-1/c mod d

a-b/c & d\*e / a-3

-a or c = d and e

a>b xor c or d <= 17

-a+b