Programming Languages: Homework #4

Due on March 12, 2019 at 9:00am

Erin Keith Section 101

Michael DesRoches

Problem 1

```
(24pts) Translate the following expression into (a)postfix and (b)prefix
notation:

  (b + sqrt(b x b − 4 x a x c))/(2 x a)
```

Solution

a)

$(b * sqrt(bb * 4ac * -)+)(2a*)/$

b)

$/(+bsqrt(- * bb * *4ac))(*2a)$

Problem 2

```
(26pts) Some languages (e.g., Algol 68) do not employ short−circuit evaluation
for Boolean expressions. However, in such languages an if ... then ... else
construct (which only evaluates the arm that is needed) can be used as
an expression that returns a value. Show how to use if...then...elseto
achieve the effect of short−circuit evaluation for A and Band for A or B.
```

Solution
For A and B, short- circuit evaluation can look like:

//return FALSE A(FALSE) ANDTH B(FALSE)
//return FALSE A(FALSE) ANDTH B(TRUE)
//return FALSE A(TRUE) ANDTH B(FALSE)
//return TRUE A(TRUE) ANDTH B(TRUE)

For A or B, short circuit evaluation can look like:

//return FALSE A(FALSE) OREL B(FALSE)
//return TRUE A(FALSE) OREL B(TRUE)
//return TRUE A(TRUE) OREL B(FALSE)
//return TRUE A(TRUE) OREL B(TRUE)

Problem 3

(24pts) Consider a midtest loop, here written in C, that processes all lines in the input until a blank line is found:

```
for (;;)
{
  line = read_line();
  if (all_blanks(line)) break;
  process_line(line);
}
```

Show how you might accomplish the same task in C using a (a) whileand(b) doloop, if breakinstructionswere not available.

Solution

```
while loop:

line = read_line();

while(!all_blank(line)){
  process_line(line);
  line = readline();
}

do loop:

do{
  line = read_line();

  if(!all_blanks(line))
  process_line(line);
}

while(!all_blanks(line));
```

Problem 4

(26pts)Write a tail−recursivefunction in Scheme to compute nfactorial
(n! = 1x2x...xn). You will probably want to define a helper function, as
discussed in the textbook.

Solution
Hepler functions make our programs easier to read

```
(define (hepler x function)
(if 0 ? x)
function
(helper(-n1)(* function x))))

(define factorial x)
(hepler x1)
```

Problem 5

(Extra Credit −10pts) Give an example in C in which an in−line
subroutine may be significantly faster than a functionally equivalent macro.
Give another example in which the macro is likely to be faster. Hint:
think about applicative versus normal−order evaluation of arguments.

Solution

```
#define someFunction(someExpression){int x = something * 5 + something - something}

inline int someFunction(someExpression){something * 3 + something - something}
```