Individual Compiler Assignment - w3

Jonas Brunsgaard

December 16, 2012

1 Exercise 1

1.1 1.a

In Call-by-value, anything passed into a function call is unchanged in the caller's scope when the function returns.

```
f(5, 2, 5){
    a = b + c              :a = 7
    b = c - b              :b = 3
    c = c + a              :c = 12
    return(7 + 3 + 12)     :+(a,b,c) = 22
}
```

thus the function will print 22,5,2.

1.2 1.b

In Call-reference, a function receives an implicit reference to a variable used as argument, rather than a copy of its value.

```
f(5, 2, 5){
    a = b + c              :a = c = 7
    b = c - b              :b = 7 - 2 = 5
    c = c + a              :c = a = 7 + 7 = 14
    return(14 + 5 + 14)    :+(a,b,c) = 33
}
```

thus the function will print 33,14,5.

1.3 1.c

In Call-value-result, is a special case of call-by-reference where the provided reference is unique to the caller. In this exercise the copies will be made in left-to-right order both on entry and on return.

```
f(5, 2, 5){
    a = b + c              :a = 5 + 2 = 7
    b = c - b              :b = 5 - 2 = 3
    c = c + a              :c = 5 + 7 = 12
    return(7 + 3 + 12)     :+(a,b,c) = 22
}
```

we copy back left to right, thus x = a

$$\begin{aligned} x &= a \\ y &= b \\ x &= c \end{aligned}$$

thus the function will print 22,12,3.

2 Exercise 2

2.1 2.a - static scoping

The call $f(4)$ will print 5

The call $f(7)$ will print 5

2.2 2.b - dynamic scoping

The call $f(4)$ will print 4

The call $f(7)$ will print 9

2.3 2.c

It could be implemented as a simple persistent symbol tables, as described in the book page 93. The *enter* and *exit* functionality does not need to be implemented. An empty list is used for the symbol table. When binding a (key, value) pair are added to front of the list. When looking up an identifier just iterate through the list. Return the value of the first matching key/value pair, if no pair if found throw an 'unbound' exception.

3 Exercise 3

3.1 3.a

Exp	Type	Unify
f	b1 * b2 * b3 -> c	
x	b1	
y	b2	
z	b3	
map	(a1 -> a2) * [a1] -> [a2]	
length	[a3]-> int	
map(length,x)	[int]	a1 = [a3] a2 = int b1 = [[a3]] = [a1]
(map(length,x),x)	[int] * [[a3]]	
length	[a4] -> int	
length(x)	int	
=	a5 * a5 -> bool	
2		
=(length(x),2)	bool	a5 = int
map	(a6 ->)a7*[a6]-> [a7]	
redint	[int] -> int	
map(redint,y)	[int]	a6 = [int] a7 = int b2 = [[int]]
(map(redint,y),z)	[int]* a8	a8 = b3
IF	bool * a9 * a9 -> a9	
IF(...)	[int]*[[a3]]	a8 = [[a3]] c = [int] * [[a3]]
x	[[a3]]	
y	[[int]]	
z	[[a3]]	
f(x,y,z)	[int]*[[a3]]	
f	[[a3]]*[[int]]*[[a3]] -> [int]*[[a3]]	

3.2 3.b

JPEG image of the solution uploaded to Absalon.