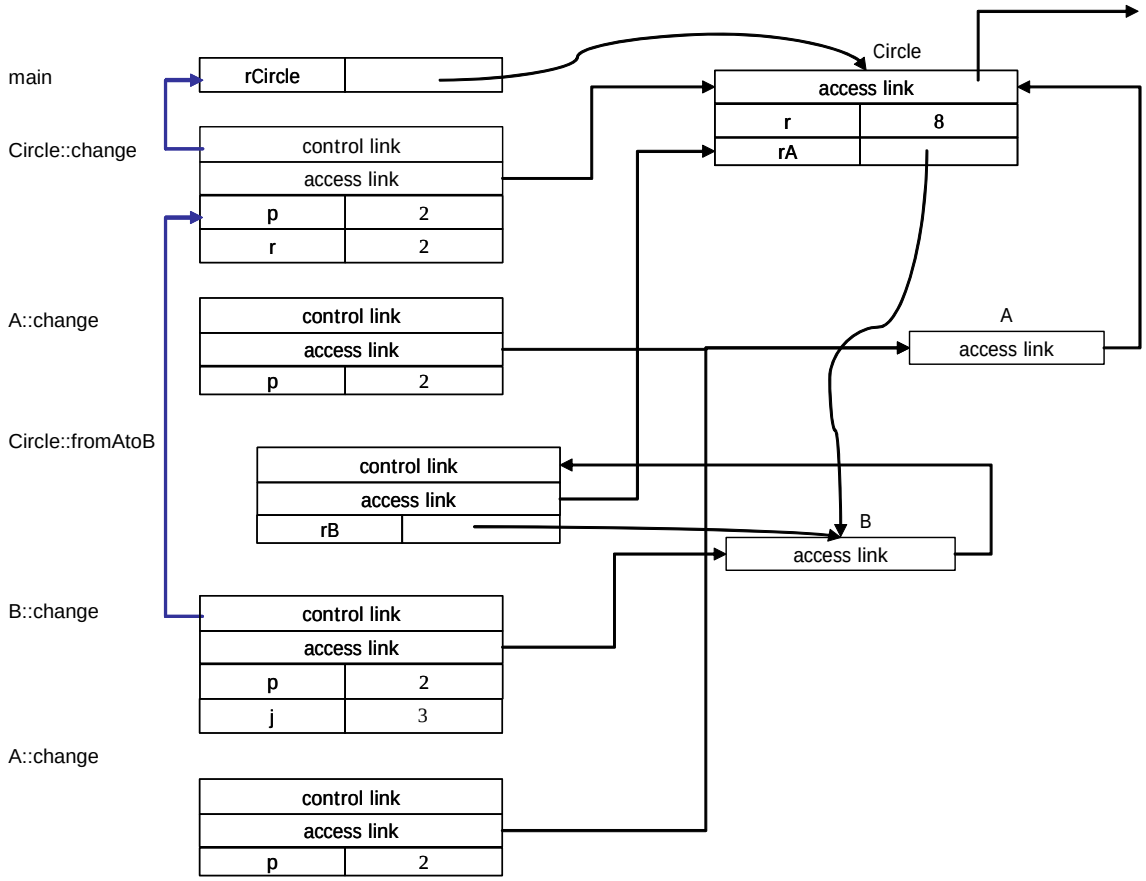
INF3110

Solutions week 5.10 - 9.10, 2009

Problem 1



The reason that the A::change activation record just below the activation record for Circle::change and the activation record for Circle::fromAtoB are not on the stack is that these have been executed before the execution of rA.change(p) marked with (‘). The A::change activation record at the bottom of the figure has been executed as part of the execution of B::change and is therefore not longer on the stack.

Problem 2

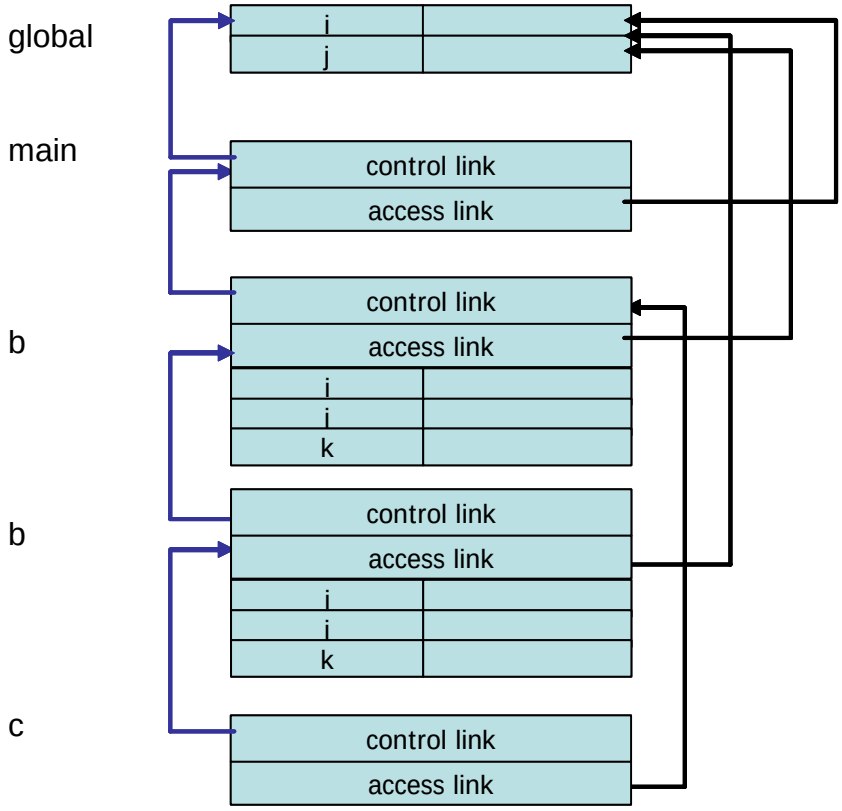
By a first glance it seems strange the the procedure Jensen is called with ix, that has not been assigned a value before the call. However, because of by-name the variable ix is assigned to within the procedure (i:= index), that is ix takes on the values from 1 to 10/5/10 for res1/res2/res3.

The x which is added to sum within the loop will for each iteration be ix*ix/a(ix)/if ...

res 1: 1*1 + 2*2 + 3*3 + ... 10*10
res 2: 7+(-1)+11+8+4

res 3: 1+1+1+0+0

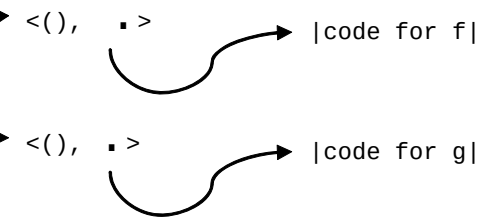
Problem 3



In order to set the access link correctly for the c-activation record, a closure has been used. The c being transferred as parameter is defined in b. There are two b-activation records on the stack, and the right one for the access link is the one that executes b(c).
The exercise did not ask for a depiction of the closure, therefore it is not part of the solution. A closure for the c transferred as parameter would be the pair consisting of the environment for c, that is the b-activation record just below main, and the code for c.

Problem 4

(1)	access	(1)
	x	5
(2)	access	(1)
	f	▪
(3)	access	(2)
	g	▪
(4)	access	(3)
	x	10
(5) g(f)	access	(2)
	h	▪
	x	7
(6) h(x)	access	(1)
	x	7



Value of expresssion : (5+7)-2 = 10. The reason is that x in (x-y)-2 denotes the global x, and that is not changed by any of the other occurrences of x, e.g. val x=7 and val x=10, as these introduce new variables.

Problem 5

NO: Imagine a variable i that is defined in an inner scope and passed as a by-reference parameter to a function f that is defined in an outer scope where i is not defined. Thereby the code of f gets access to the L-value of i. The only requirement is that i is visible in the block that contains the call f(i).

```
{    -- block that does not contain i
  void f(ref int j) {
    ... j= ...
  }
  ...
  { int i;
    f(i)
  }
}
```

Problem 6

At least two cases:

1. If it accesses global variables
2. If it reads values

Problem 7

By reference: p(x) will have the effect that x gets the value 3. The reason is that the assignment to i in p also assigns to x.

By value-result: p(x) will have the effect the x gets the value 2. The local i (in the p activation record) will get the value 1, then 2 (by the assignment). The assignment to x will assign 2 to x. Upon return, the value of the local i (2) will be assigned to x.

Problem 8

For the case where an expression is used as an actual parameter to a name parameter, the name parameters may be represented by a function. This function has to be defined in the same scope as the call in order to get access to the variables that are part of the expression.

This will not cover the case where a name parameter is assigned to.

Problem 9

a) a = p(a)

b) a = p(a)

c) That is not as simple as above? a and b have to be assigned to two attributes of an object, this object is used as actual parameter, p has to change these variables, and upon return the values of these two attributes have to be assigned to a and b.