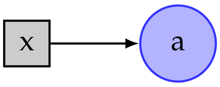
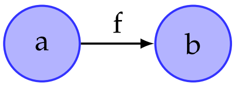
```
def traverseSummarizedList(x: AcyclicList) = {
// variable which afterwards references
// the last element
var end : AcyclicList = null

var cur = x
while (cur != null) {
end = cur
cur = cur.n
}
}
```

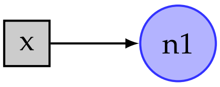
Listing 1: List traversal testcase

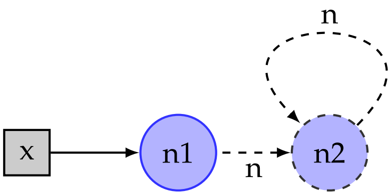


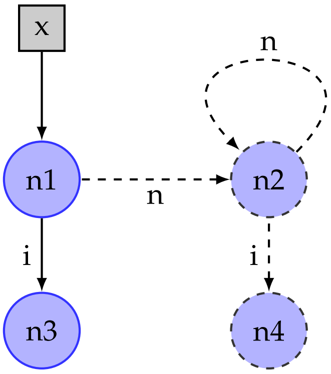


```
def createList(n: Int) = {
  var x = new E
  var t: E = null

  var i = 1
  while (i < n) {
    t = new E
    t.f = x
    x = t
    t = null
    i += 1
  }
}
```

Listing 2: List traversal testcase







(a) Structure 1

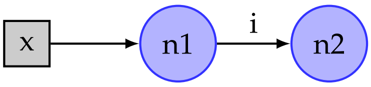
Abbr.	Full ID	Semantic Domain
n1	L194C17_0	$\top$
n2	L196C17_0+L198C17_0	$\top$
n3	L195C15_0	[0..0]
n4	L197C15_0+L199C15_0	[0..0]
x	x	$\top$
sum	sum	[0..0]

(b) Heap IDs and semantic state

Figure 1: Summing up list elements: Result

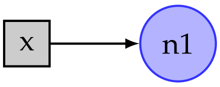
```
def iAssign(unknown:Boolean) = {
  val x = new IntNode

  if (unknown) {
    x.i = 1
  } else {
    x.i = 2
  }
}
```

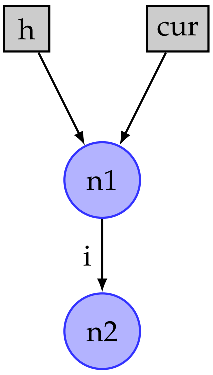
Listing 3: wbla

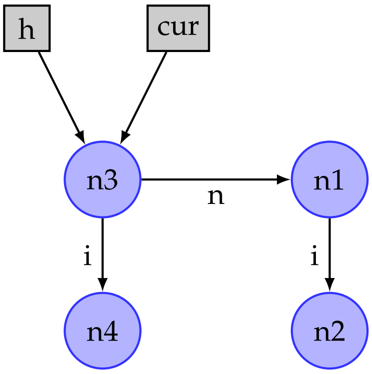


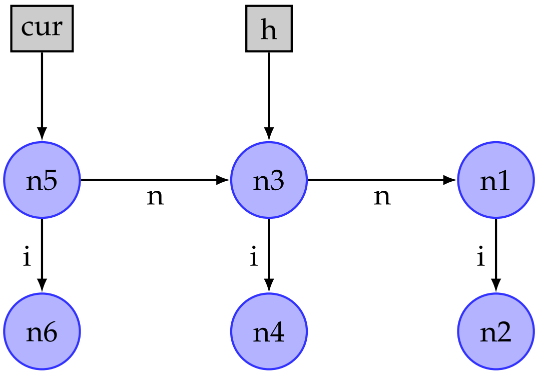
```
def sumListElementsZero = {
  x = /* code to build and initialize
         a list with fields set to 0 */

  var cur = x
  var sum = x.i

  while (cur != null) {
    sum += cur.i
    cur = cur.n
  }
}
```

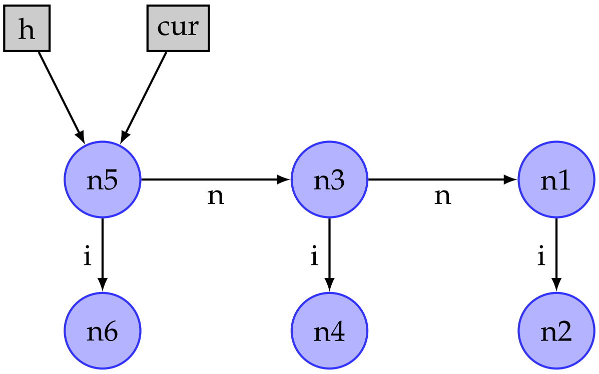
Listing 4: Sum list elements testcase

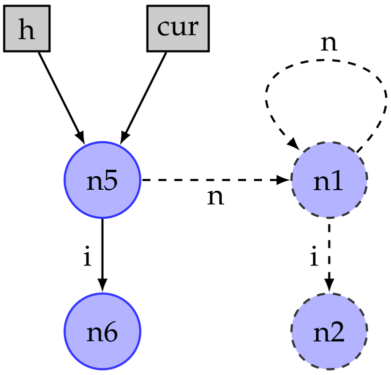



h
cur
n1
i
n2


h
cur
n3
n
n1
i
i
n4
n2


cur
h
n5
n
n3
n
n1
i
i
i
n6
n4
n2


h
cur
n5
n
n3
n
n1
i
i
i
n6
n4
n2



```
def createZeroList(n: Int) = {
  var h: IntNode = null
  var cur :IntNode = null
  var i = 0

  while (i < n) {
    cur = new IntNode
    cur.i = 0
    cur.n = h
    h = cur
    i += 1
  }
}
```