COMP0020: Functional Programming

Lecture 15

Lists, Trees and Graphs

(link to implementation issues)

Contents

- Representation in memory : LISTS
- Trees
  - Representation in memory
  - Multiway branching trees
- Graphs
  - Source code
  - Representation in memory

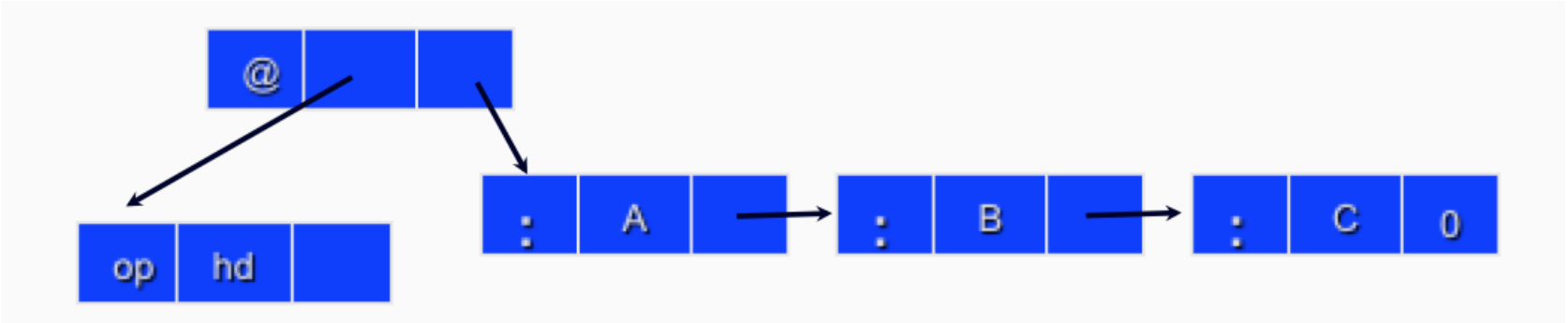
How lists are represented in memory

We cover basic mechanisms : in practice, functional language implementations (such as Miranda) may use different representations in different contexts (e.g. whether the list is a data value embedded in the program and known at compile-time, or whether it is consructed dynamically at run-time).

A very simple strategy is to manage physical memory locations in groups of three (called "cells") :

- A "tag" (indicating the kind of cell) followed by two other items ("fields")
- Can be anywhere in memory

x = ('A' : ('B' : ('C' : [])))
main = hd x



Representing TREES in memory

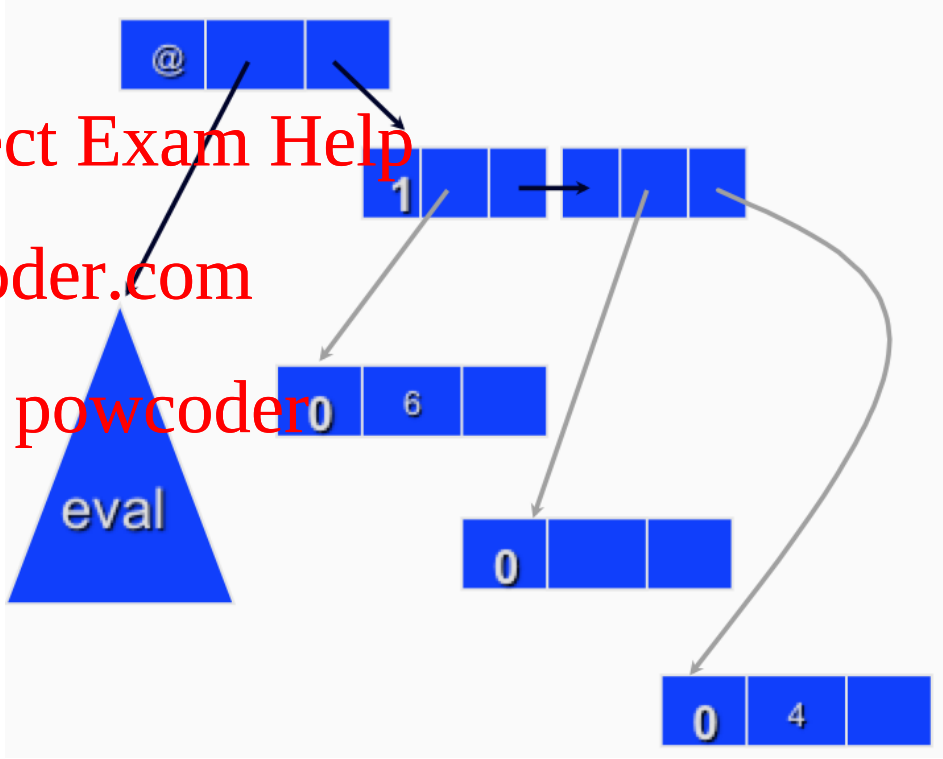
exp ::= *Const num*
| *App exp op exp*
| *Bracketed exp*

op ::= *Plus* | *Minus* | *Times* | *Divide*

x :: *exp*

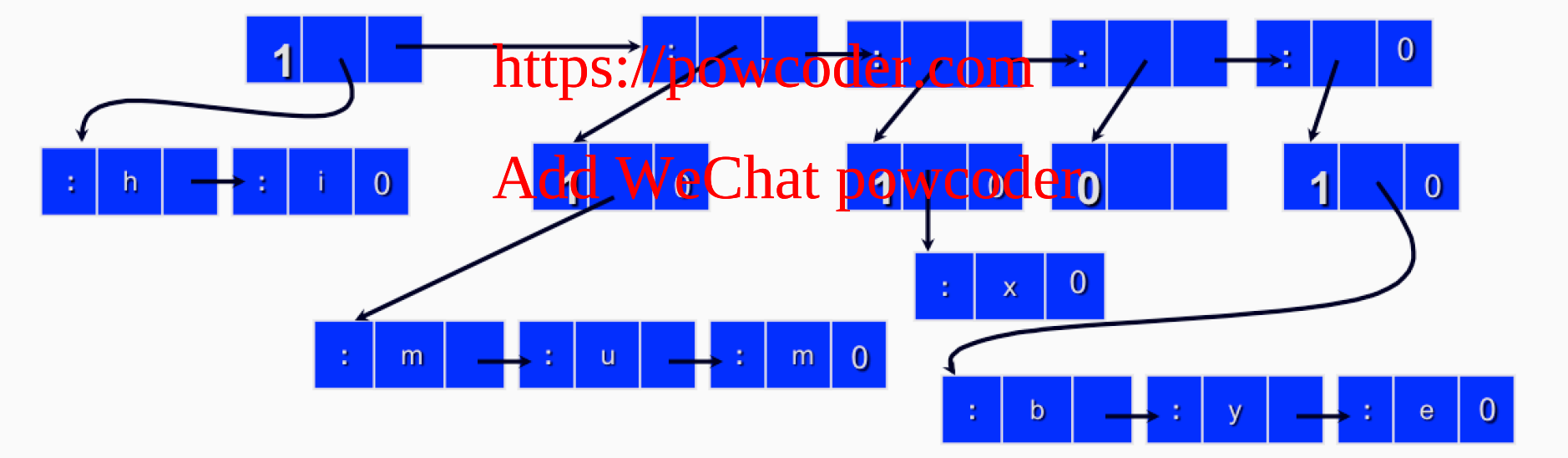
x = *App* (*Const* 6) *Plus* (*Const* 4)

main = *eval x*



Multiway branching trees

multitree * : := Empty | Node * [multitree *]
x : : multitree [char]
x = Node “hi" [(Node “mum" []), (Node “x" []), Empty, (Node “bye" [])]



Multiway branching trees

multitree * : := Empty | Node * [multitree *]

x : : multitree [char]

x = Node “hi" [(Node “mum" []), (Node “x" []), Empty, (Node “bye" [])]


```
rightmost :: multitree * -> *
rightmost Empty          = error "rightmost of empty tree"
rightmost (Node x [])    = x
rightmost (Node x ns)    = rightmost (last ns1), if (ns1 ~= [])
                         = x, otherwise
                           where
                           ns1 = filter (~= Empty) ns
```

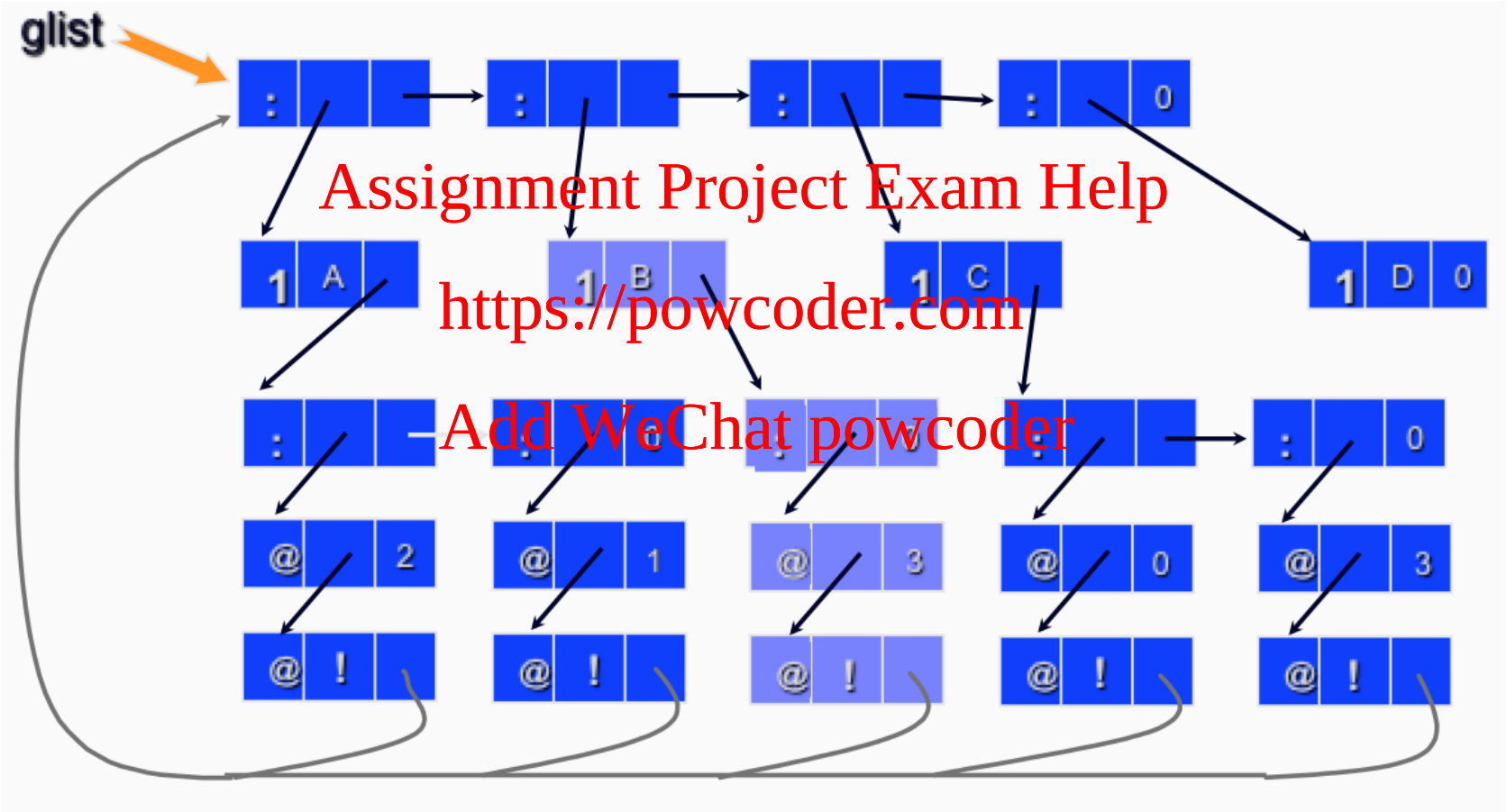
Graphs : simple source code

```
multitree* ::= Empty      | Node  * [multitree *]      || (as before)
a  =       Node 'A'   [c, b ]
b  =       Node 'B'   [d]
c  =       Node 'C'   [a, d ]
d  =       Node 'D'   []
graph ::   multitree char
graph =    a
firstlink  :: multitree *      -> multitree *
firstlink  Empty               = error "firstlink of empty graph"
firstlink  (Node x [])         = error "firstlink of node with no first link"
firstlink  (Node x (n : ns))   = n
```

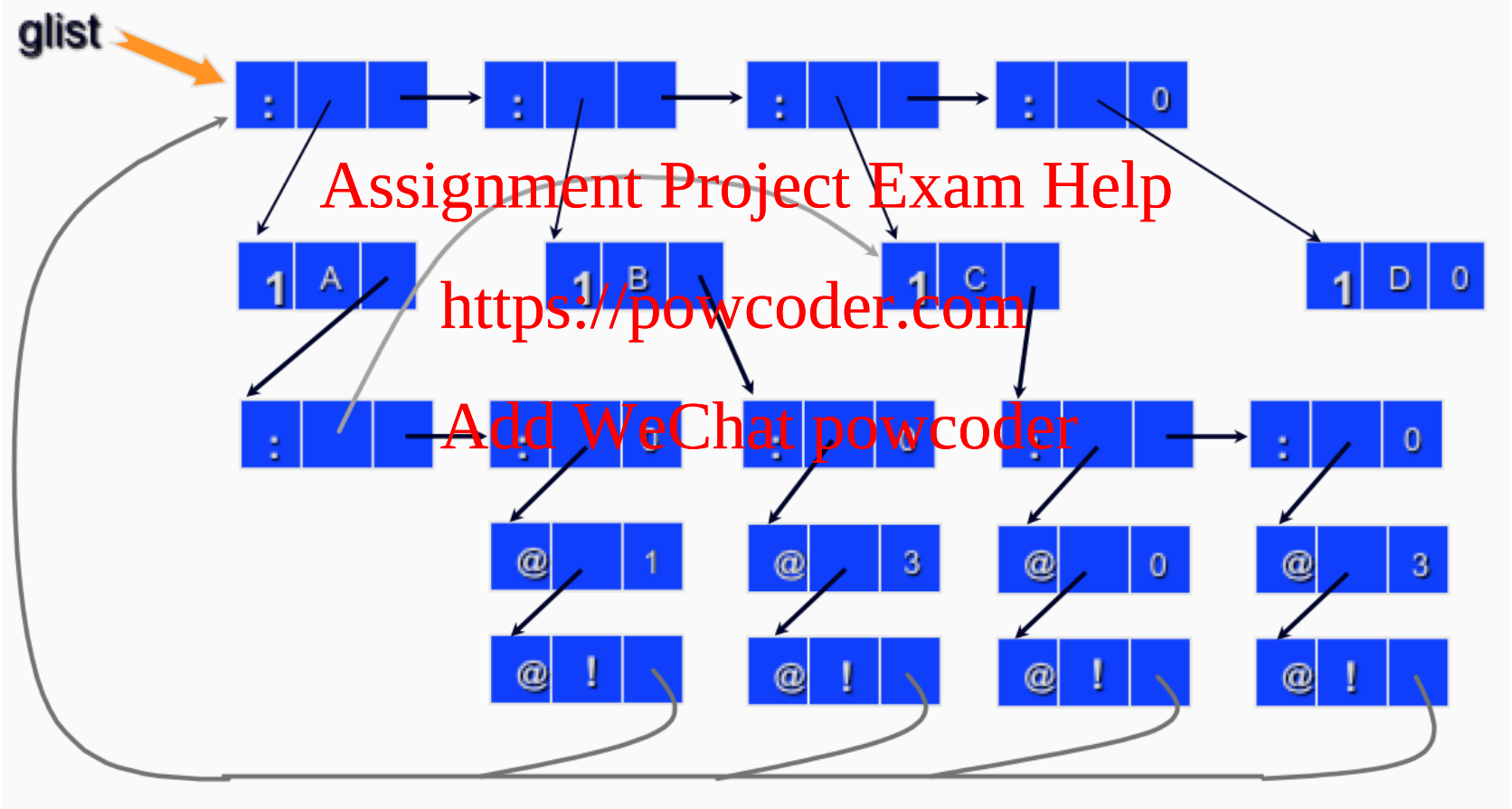
Graphs : source code using a list

```
multitree *  ::=  Empty     |  Node  *  [multitree *]        ||  (as before)
glist  ::   [multitree char]
glist  =    [ Node 'A'  [  (glist ! 2), (glist ! 1) ],
              Node 'B'  [   (glist ! 3) ],
              Node 'C'  [  (glist ! 0), (glist ! 3) ],
              Node 'D'  []  ]
graph ::   multitree char
graph =    hd glist
firstlink   ::  multitree *     -> multitree *
firstlink   Empty               =  error "firstlink of empty graph"
firstlink   (Node x [])         =  error "firstlink of node with no first link"
firstlink   (Node x (n : ns))   =  n
```

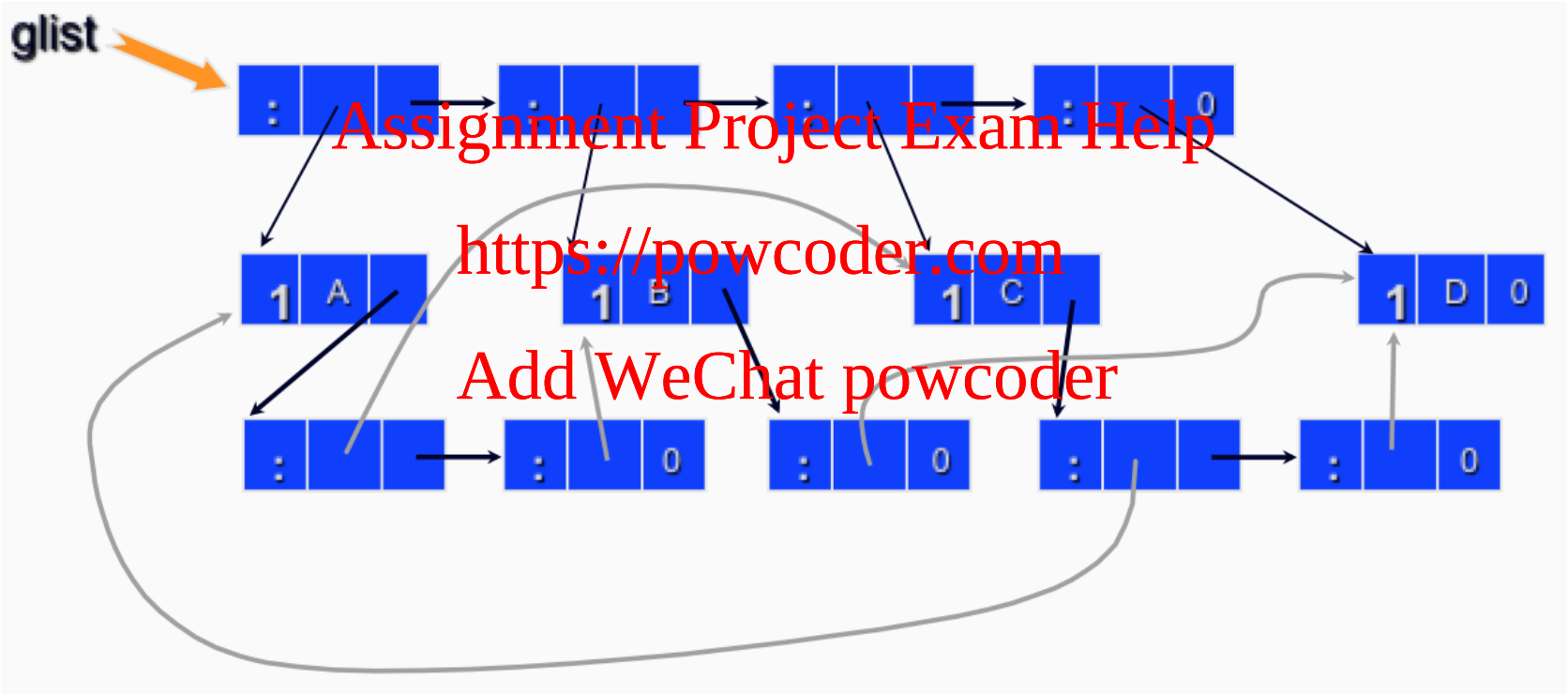
Graphs : representation



Graphs : representation after evaluating (firstlink graph)



Graphs : representation after evaluating all links



Summary

- Representation in memory : LISTS
- Trees
  - Representation in memory
  - Multiway branching trees
- Graphs
  - Source code
  - Representation in memory

END OF LECTURE