```
delete pseudocode
set CLEANUP and isSimpleDelete flags to false
while(true)
{
  do primarySeekForDelete(deleteKey)
  if node->secDoneFlag is set
    help
    continue from top of primary seek's while loop
  if(!CLEANUP)
  {
    try CAS(node->lChild,<nlChildAddr,0,0>,<nlChildAddr,1,0>)
    if CAS FAILED
      help
      continue from top of primary seek's while loop
    if CAS SUCCEEDED
      set CLEANUP to true and set storedNode = node
  }
  if(storedNode != node) //Someone removed the node for me. So DONE
  set "deleteFlag" on node's rChild using BTS
  if complex delete
  {
    while(true) //secondary seek
    {
      nRChild = node->rChild
      if(nRChild != NULL)
        isSplCase = secondarySeekForDelete(nRChild)
      else
        set isSimpleDelete flag to true
        break from while loop
      if node key is unmarked
      {
        try CAS(rnode->lChild,<NULL,0,0>,<nodeAddr,0,1>)
        if CAS failed
        {
          if promoteFlag is set
            if address does not match with node's address
              restart primary seek. assert(node->secFlag == DONE)
              break from while loop
          else
            if address != NULL //restart secondary seek
              continue from top of secondary seek's while loop
            else
              assert(rnode->lChild's deleteFlag is set)
              help operation at secondaryLastUnmarkedEdge
              if secondaryLastUnmarkedEdge does not exist, then help node->rChild
              //simplehelp(node,nrChild)
              continue from top of secondary seek's while loop
        }
        set promote flag on rnode->rChild using BTS
        promote key using a simple write. Node's key changes from <0,kN> to <1,kRN>
      }
      if(!isSplCase)
      {
        try CAS(rpnodeLChild,<rnode,0,0>,<rnodeRChild,0,0>) //remove secondary node
        if CAS FAILED, help operation at secondaryLastUnmarkedEdge
```

```
55              if secondaryLastUnmarkedEdge doesn't exist, override CASinvariant and help
                node->rChild  //simplehelp(node,nrChild)
56              continue from top of secondary seek's while loop
57            if CAS SUCCEEDED, set node->secDoneFlag to true
58          }
59          else
60          {
61              try CAS(nodeRChild,<rnode,1,0>,<rnodeRChild,1,0> //no problem if CAS fails
62              set node->secDoneFlag to true
63          }
64          oldNodeAddr = address of node
65          while(true)
66          {
67            create a fresh copy of node
68            newNodeKey as <0,kRN>
69            newNodeLChild as <node's lChildAddr,0,0>
70            newNodeRChild = <node's rChildAddr,0,0>
71            try CAS(pnode->lChild,<node,0,0>,<newNode,0,0>)
72            if CAS SUCCEEDED then DONE
73            if CAS FAILED
74              if address has changed
75                then someone helped me install a fresh copy.so DONE
76              else
77                CAS has failed coz the edge is marked.
78                if lastUnmarkedEdge is not (pnode,node) then help
79                do primarySeekForDelete(node->key) //restart primary seek with new key
80                if the new key is not found then someone has installed a fresh copy. So done

81                if key is found and newNodeAddr != oldNodeAddr then someone has installed a
                  fresh copy. So done
82          }
83        }
84      }
85      else //simple delete
86        set isSimpleDelete to true
87      if(isSimpleDelete)
88      {
89        try CAS(pnode->lChild,<node,0,0>,<node's l/r child,0,0>)
90        if CAS SUCCEEDED, then DONE
91        if CAS FAILED
92          if lastUnmarkedEdge is NOT (pnode,node) help
93      }
94    }
95    //simplehelp(node,node's rChild)
96    simplehelp(pnode,node)
97    assert(pnode's secDoneFlag not set)
98    set delete flag on node->rChild using BTS
99    if complex delete
100     if node->secDoneFlag is set
101       create a fresh copy of node
102       try CAS(pnode->rChild,<node,1,0>,<newNode,1,0>)
103   else //simple delete
104     try CAS(pnode->rchild,<node,1,0>,<node's lchild,1,0>)
```