```
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
```

```
          if secondaryLastUnmarkedEdge doesn't exist, override CASinvariant and help
          node->rChild  //simplehelp(node,nrChild)
          continue from top of secondary seek's while loop
        if CAS SUCCEEDED, set node->secDoneFlag to true
      }
      else
      {
          try CAS(nodeRChild,<rnode,1,0>,<rnodeRChild,1,0> //no problem if CAS fails
          set node->secDoneFlag to true
      }
      oldNodeAddr = address of node
      while(true)
      {
        create a fresh copy of node
        newNodeKey as <0,kRN>
        newNodeLChild as <node's lChildAddr,0,0>
        newNodeRChild = <node's rChildAddr,0,0>
        try CAS(pnode->lChild,<node,0,0>,<newNode,0,0>)
        if CAS SUCCEEDED then DONE
        if CAS FAILED
          if address has changed
            then someone helped me install a fresh copy.so DONE
          else
            CAS has failed coz the edge is marked.
            if lastUnmarkedEdge is not (pnode,node) then help
            do primarySeekForDelete(node->key) //restart primary seek with new key
            if the new key is not found then someone has installed a fresh copy. So done

            if key is found and newNodeAddr != oldNodeAddr then someone has installed a fresh
            copy. So done
      }
    }
  }
  else //simple delete
    set isSimpleDelete to true
  if(isSimpleDelete)
  {
    try CAS(pnode->lChild,<node,0,0>,<node's l/r child,0,0>)
    if CAS SUCCEEDED, then DONE
    if CAS FAILED
      if lastUnmarkedEdge is NOT (pnode,node) help
  }
}
//simplehelp(node,node's rChild)
simplehelp(pnode,node)
assert(pnode's secDoneFlag not set)
set delete flag on node->rChild using BTS
if complex delete
  if node->secDoneFlag is set
    create a fresh copy of node
    try CAS(pnode->rChild,<node,1,0>,<newNode,1,0>)
else //simple delete
  try CAS(pnode->rchild,<node,1,0>,<node's lchild,1,0>)
```