ASSIGNMENT-4 Amandeep Kaur

Main.c

```
----------------------------- Hoare Test-----------------------------------
--
--Testing the hoare semantics with the help of monitor class
--There is one monitor and 2 CV
--


var
     mon:mytest
function Hoaretest()
 var

--create array of new threads
     tharray:array[4] of Thread=new array of Thread{4 of new Thread}
     mon=new mytest
     mon.Init()                                  --initiate mutex
     print("Intialize Tester")
     printChar('\n')

   --initialize threads
     tharray[0].Init("Threads-1")
     tharray[0].fork(threadtest1,0)
     tharray[1].Init("Threads-2")
     tharray[1].fork(threadtest1,1)
     tharray[2].Init("Threads-3")
     tharray[2].fork(threadtest2,2)
     tharray[3].Init("Threads-4")
     tharray[3].fork(threadtest1,3)

ThreadFinish()
endFunction

function threadtest1(i:int)
  mon.customer(i)
  currentThread.Yield()
  mon.producer(i)
  currentThread.Yield()
endFunction

class mytest                              --monitor class
   superclass Object
fields
    const BUFFER_SIZE=5
    buffer:array[BUFFER_SIZE] of char
    in,out:Semaphore
    count:int
    mutex1:Mutex
    threadwait,threadfree:HoareCondition

methods
     Init()
     customer(i:int)
     producer(i:int)
  printbuf()
endClass

behavior mytest

method Init()
   threadwait=new HoareCondition           --initialize variables
   threadwait.Init()
   mutex1=new Mutex
   mutex1.Init()
endMethod

method customer(i:int)          --monitor method to see how wait and signal works
    mutex1.Lock()
    print("Customer start:\n")
    print(currentThread.name)
    if(count==10)
      print("Customer name before wait:\n")
      print(currentThread.name)
      threadwait.Wait(&mutex1)
      print("Customer name after wait:\n")
      print(currentThread.name)
    endIf
    buff[in]=c
    in=(in+1) % 10
    count=count+1
    threadfree.Signal(&mutex1)
    print("Customer name after signal:\n")
    print(currentThread.name)
    mutex1.Unlock()
endMethod

method producer(i:int)            --monitor method to see how wait and signal works
      mutex1.Lock()
     print("Producer start:\n")
     print(currentThread.name)
    if(count==10)
      print("Producer name before wait:\n")
      print(currentThread.name)
      threadwait.Wait(&mutex1)
      print("Producer name after wait:\n")
      print(currentThread.name)
    endIf
    c=buff[out]
    out=(out+1) % 10
    count=count-1
    threadfree.Signal(&mutex1)
    print("Producer name after signal:\n")
   print(currentThread.name)
    mutex1.Unlock()
endMethod

method printbuf(c:char)                       --print buffer
  var
    i,j:int
   mutex1.Unlock()
   print(" ")
   print(currentThread.name)
   print(c)
   printChar('\n')
   j=out
   for i=1 to count
    printChar(buff[j])
   j=(j+1)%10
   endFor
    for i=1 to 10-count
    printChar(' ')
   endFor
   mutex1.Unlock()
endMethod


endBehavior
```