```
-----------------------Gaming Parlor---------------------
var
   monitor1: GameParlor
   group: array [8] of Thread = new array of Thread{ 8 of new Thread}

function GamingParlor()

  monitor1= new GameParlor
  monitor1.Init()
  group[0].Init("A")
  group[0].Fork(groupre,4)
  group[1].Init("B")
  group[1].Fork(groupre,4)
  group[2].Init("C")
  group[2].Fork(groupre,5)
  group[3].Init("D")
  group[3].Fork(groupre,5)
  group[4].Init("E")
  group[4].Fork(groupre,2)
  group[5].Init("F")
  group[5].Fork(groupre,2)
  group[6].Init("G")
  group[6].Fork(groupre,1)
  group[7].Init("H")
  group[7].Fork(groupre,1)
  --ThreadFinish()

endFunction

function groupre(g:int)
var
  i:int
for i=1 to 5
  monitor1.Request(g)
  currentThread.Yield()
  monitor1.Return(g)
  currentThread.Yield()
endFor
endFunction

class GameParlor
 superclass Object
fields
  monitorLock: Mutex
  numberDiceAvail:int
  groupwait:Condition
  front1:Condition
  --myList:List[Thread]
  numwaitgrp:int
methods
  Init()
  Request(numberOfDice:int)
  Return(numberOfDice:int)
  Print(str: ptr to array of char,count: int)
endClass

behavior GameParlor

method Init()
  numberDiceAvail=8
  --myList= new List[Thread]
   groupwait=new Condition
  front1=new Condition
monitorLock = new Mutex
  monitorLock.Init()
  groupwait.Init()
  front1.Init()
endMethod

method Request(numberOfDice:int)
 --var
    --p:ptr to Thread
```

```
  monitorLock.Lock()
     self.Print("requests",numberOfDice)
     --myList.AddToEnd(currentThread)
     --while currentThread!=
         --myList.Remove(p)
     numwaitgrp=numwaitgrp+1
     if numwaitgrp > 1
     groupwait.Wait(& monitorLock)
     endIf
     while (numberOfDice > numberDiceAvail)
        front1.Wait(& monitorLock)
     endWhile
     --endWhile
     numberDiceAvail= numberDiceAvail-numberOfDice
     numwaitgrp=numwaitgrp-1
      groupwait.Signal(& monitorLock)
       --myList.AddToFront(p)
     self.Print("proceeds with",numberOfDice)
  monitorLock.Unlock()
endMethod


method Return(numberOfDice:int)
  monitorLock.Lock()
     numberDiceAvail= numberDiceAvail+numberOfDice
     front1.Signal(& monitorLock)
     --myList.Remove(currentThread)
     self.Print("releases and adds back",numberOfDice)
  monitorLock.Unlock()
endMethod

method Print(str :ptr to array of char, count: int)
print(currentThread.name)
print(" ")
print(str)
print(" ")
printInt(count)
printChar ('\n')
print("-------------------Number of Dice now avail= ")
printInt(numberDiceAvail)
printChar ('\n')
endMethod

endBehavior

endCode
```