**Jacob Alspaw**

**EECS 338**

**Assignmnet 4**

**Assumptions:**

* Everyone is standing in a single line.
* No preference customers count as a potential customer for Judy and for Tina, and will therefore be included in any customer type’s consideration for leaving the store. A Tina customer will not leave if Judy has a line of customers that exceeds five, and vice versa.

**Monitor B {**

//variable declaration

Conditions: Tina, Judy

Integers: TinaOnlyPotentialsInLine, JudyOnlyPotentialsInLine

Booleans: isTinaBusy, isJudyBusy

Lists: customers

//intiliazation

begin

TinaOnlyPotentialsInLine = 0 //initially no tina customers in line

JudyOnlyPotentialsInLine = 0 //initially no judy customers in line

isTinaBusy = false //tina is not initially busy

isJudyBusy = false //judy is not initially busy

customers = {} //the line is initially empty

end

**Procedure customer TinaBarrier()** {

Tina.wait()

customer = findFirstTinaCustomerInLine // Remove 1st Tina-Only or Don’t-Care in line

if (customer == Don’t-Care) {

JudyOnlyPotentialsInLine--

}

TinaOnlyPotentialsInLine--

return customer

}

**Procedure void TinaChecksLine ()** {

isTinaBusy = False

if (TinaOnlyPotentialsInLine > 0) { // if more customers in line, Tina cuts

isTinaBusy = True

Tina.signal()

}

}

**Procedure customer JudyBarrier()** {

Judy.wait()

customer = findFirstJudyCustomerInLine // Remove 1st Judy-Only or Don’t-Care in line

if (customer == Don’t-Care) {

TinaOnlyPotentialsInLine--

}

JudyOnlyPotentialsInLine--

return customer

}

**Procedure void JudyChecksLine ()** {

isJudyBusy = False

if (JudyOnlyPotentialsInLine > 0) { // if more customers in line, Judy cuts

isJudyBusy = True

Judy.signal()

}

}

**Procedure void TinaCustomerEnters()** {

If (TinaOnlyPotentialsInLine < 5) {

customers.add(this) // adds customer to end of line (this is the customer) TinaOnlyPotentialsInLine++

If (! isTinaBusy) {

isTinaBusy = true

Tina.signal()

}

} else {

customer leaves

}

}

**Procedure void JudyCustomerEnters()** {

If (JudyOnlyPotentialsInLine < 5) {

customers.add(this) // adds customer to end of line (this is the customer) JudyOnlyPotentialsInLine++

If (! isJudyBusy) {

isJudyBusy = true

Judy.signal()

}

} else {

customer leaves

}

}

**Procedure void DontCareCustomerEnters()** {

if (customers.length < 5) {

customers.add(this) // adds customer to end of line

TinaOnlyPotentialsInLine++

JudyOnlyPotentialsInLine++

If (! isTinaBusy) { // check Tina, then Judy

isTinaBusy = true

Tina.signal()

} else if (! isJudyBusy) {

isJudyBusy = true

Judy.signal()

} else {

customer leaves

}

}

} //end montior

**Process List**

**Tina-Process:**

while (true) {

c = B.TinaBarrier() // until a customer is returned, Tina sleeps

cuthair(c)

B.TinaChecksLine()

}

**Judy-Process:**

while (true) {

c = B.JudyBarrier() // until a customer is returned, Judy sleeps

cuthair(c)

B.JudyChecksLine()

}

**Tina-Only:**

B.TinaCustomerEnters()

**Judy-Only:**

B.JudyCustomerEnters()

**Don’t-Care:**

B.DontCareCustomerEnters()

**Explanations:**

**Tina Explanation:** Tina is always cutting hair. We wait for Tina to be needed. We wait for access to mutually exclusive variables (the line) which are used in the monitor’s procedure for Tina. We find, remove, and remember the next customer in the single line that will let Tina cut their hair. We decrement number of people in line that will let Tina cut their hair. If that person didn’t care who cut their hair, then we must also decrease the amount of people that would let Judy cut their hair. We stop accessing mutually exclusive variables (the line) in the monitor because the procedure has ended. We cut the persons hair. Tina is no longer busy because she has finished cutting that person’s hair. We wait for access to shared variables in the monitor, so Tina checks the line to see if anyone else needs their hair cut. If someone is in line that needs their hair cut, Tina will tell herself to repeat the process above. Otherwise, Tina will go back to sleeping.

**Judy Explanation:** Judy is always cutting hair. We wait for Judy to be needed. We wait for access to mutually exclusive variables (the line) which are used in the monitor’s procedure for Judy. We find, remove, and remember the next customer in the single line that will let Judy cut their hair. We decrement number of people in line that will let Judy cut their hair. If that person didn’t care who cut their hair, then we must also decrease the amount of people that would let Judy cut their hair. We stop accessing mutually exclusive variables (the line) in the monitor because the procedure has ended. We cut the persons hair. Judy is no longer busy because she has finished cutting that person’s hair. We wait for access to shared variables in the monitor, so Judy checks the line to see if anyone else needs their hair cut. If someone is in line that needs their hair cut, Judy will tell herself to repeat the process above. Otherwise, Judy will go back to sleeping.

**Tina-Only Customer Explanation:** Tina only customers will enter the store and wait to access shared variables in the monitor. The customer will check if the number of customers already in line that will let Tina cut their hair exceeds 5. If so, then the customer will leave the store. If the worst-case wait time is 5 or less people, then the customer will add themselves to the back of the line. If Tina is not busy (meaning she is sleeping), then the customer will notify Tina of his presence. The customer will stop access to shared variables and leave the monitor.

**Judy-Only Customer Explanation:** Tina only customers will enter the store and wait to access shared variables in the monitor. The customer will check if the number of customers already in line that will let Tina cut their hair exceeds 5. If so, then the customer will leave the store. If the worst-case wait time is 5 or less people, then the customer will add themselves to the back of the line. If Tina is not busy (meaning she is sleeping), then the customer will notify Tina of his presence. The customer will stop access to shared variables and leave the monitor.

**Don’t Care Customer Explanation:** No preference customers will enter the store and wait to access shared variables inside the monitor. The customer will check if the number of customers already in line exceeds 5. If so, then the customer will leave the store. If there are 5 or less people waiting to get their haircuts, then the customer will add themselves to the back of the line. If Tina is not busy (meaning she is sleeping), then the customer will notify Tina of his presence. He will check with Tina first, because Tina owns the store. If Tina is busy, but Judy is not, then the customer will notify Judy of his presence. The customer will stop access to shared variables and leave the monitor once in line or being waited on, whichever comes first.