Semaphores

Name	Purpose	Initial Value
readMovies	Make the movies list and the movie file accessible / modifiable by only 1entty at a time	1
availableAgent	Control access to the box office agents	0
getTransactioninfo	Control access to the agentQueue which passes info between agent and customer	1
cust_ready	Signal whether customer has enqueued their ticket choice and is ready to buy the ticket	0
soldOutRespoonse [ ]	Signal when response for a customer (given by the index of this array) regarding the sold out status of the ticket has been recorded in the shared soldOut boolean array	Array [NUM_CUSTOMERS], each one with initial value = 0
buyTicket [ ]	Signal when the box office agent has processed the ticket for a speific customer (given by the inddex of the array)	Array [NUM_CUSTOMERS], each one with initial value = 0
availableTT	Control access to the ticket taker	1
cust_ready_tt	Signal when the customer has queued their ticket and is ready to have their ticket torn	0
tore_ticket [ ]	Signal when the ticket for a specific customer (given by the index of this array) has had their ticket torn	Array [NUM_CUSTOMERS], each one with initial value = 0
availableCS	Control access to the concession stand worker	1
cust_ready_cs	Signal when customer has queued their snack order and is ready to be served	0
orderFilled [ ]	Signal when an order for a customer has been filled	Array [NUM_CUSTOMERS], each one with initial value = 0
leftTheater	Tracks whether specific	Array of Semaphores which

	customers (represented as semaphores in the array) have joined the main function again	each have 0 as initial value

Function Pseudocode

```
Main(){
        queue agentQueue
        queue ttQueue
        queue csQueue
        boolean [NUM_CUSTOMERS] soldOut
        ArrayList<Movie> movies

}

void Customer(){
        int id
        TicketTransaction ticket;
        SnackTransaction snack;

        anounceBirth()
        wait(readMovies)
        readMoviesFile()
        signal(readMovies)
        movie = decideOnMovie()
        if (soldOut(movie))
                anounceLeft()
                return
        getInLine()
        wait(availableAgent)
        wait(getTransactionInfo)
        agentEnqueue(ticket)
        signal(cust_ready)
        signal(getTransactionInfo)
        wait(soldOutResponse[id])
        if (soldOut[id])
                anounceLeft()
                signal(readMovies)
                return
        else
                wait(buyTicket[id])

        getInTicketTackerLine()
        wait(availableTT)
        //seeTicketTaker()
        ttEnqueue(ticket);
        signal(cust_ready_tt)
        wait(tore_ticket[id])
```

```
        if (gettingConcessions()){
            snack = decideOnType()
            getInConcessionStandLine()
            wait(availableCS)
            csEnqueue(snack)
            signal(cust_ready_cs)
            wait(orderFilled[id])
        }
        announceJoined()
}

void BoxOfficeAgents(){
        int id
        TicketTransaction ticket;
        boolean isSoldOut;

        announceBirth()
        while(true){
            signal(availableAgent)
            wait(cust_ready)
            wait(get_transaction_info)
            agentDequeue(ticket)
            signal(get_transaction_info)
            getTransactionInfo()
            wait(readMovies)
            if (isSoldOut()){
                    signal(readMovies)
                    soldOut[custId] = true
                    signal(soldOutResponse[id])
                    continue
            }else{
                signal(soldOutResponse[id])
                ticketSaleAccounting()
                signal(readMovies)
                sellTicket() <----sleeps then prints that it sold the ticket
                signal(buyTicket[id])
            }

        }
}

void TicketTacker(){
        TicketTransaction currTicket;
        while(true){
            wait(cust_ready_tt)
            ttDequeue(currTicket)
            tearTicket()
            signal(tore_ticket[currTicket.ownerId])
            signal(availableTT)
```

```
        }
}

void ConcessionStandWorker(){
        SnackTransaction currSnack
        while(true){
            wait(cust_ready_cs)
            csDequeue()
            announceOrderTaken()
            makeAndServeOrder()
            signal(orderFilled[currSnack.custId])
            signal(availableCS)
        }
}
```