```python
# -*- coding:iso-latin-1 -*-
import random
import threading
import time
import os

def show(mess):
    os.write(1, mess + "\n")

def down(sem):
    sem.acquire()

def up(sem):
    sem.release()

acquire = wait = down
release = signal = up

class Inspection():

    def __init__(self):
        self.passed = False
        self.requested = threading.Semaphore(0)
        self.finished = threading.Semaphore(0)
        self.lock = threading.Semaphore(1)

class Line():

    def __init__(self, customers_count):
        self.number = 0
        self.requested = threading.Semaphore(0)
        self.customers = [
            threading.Semaphore(0)] * customers_count
        self.lock = threading.Semaphore(1)

def do_inspection(inspected_count):

    mess = "manager: inspection %d" % inspected_count
    show(mess)
    return random.choice(range(approved_rate + 1)) > 0

def manager(tot_cones, approved_rate, inspection):

    approved_count = 0
    inspected_count = 0
    while approved_count < tot_cones:

        wait(inspection.requested)
        inspection.passed = do_inspection(inspected_count)
        inspected_count += 1
        if inspection.passed:
            approved_count += 1
        signal(inspection.finished)

    time.sleep(1)
    mess = "manager leaves: %d approved %d rejected" % (
        approved_count, inspected_count - approved_count)
    show(mess)

def make_cone(clerk, cone, customer):

    mess = "clerk %2d: making cone %d for customer %d" % (
        clerk, cone, customer)
    show(mess)

def log_inspection(clerk, cone, customer, passed):

    mess = "clerk %2d: cone %d for customer %d %s" % (
        clerk, cone, customer,
        ["REJECTED", "passed"][passed])
    show(mess)

def clerk(id, customer, cone, clerk_done, inspection):

    passed = False
    while not passed:
        make_cone(id, cone, customer)
        acquire(inspection.lock)
        signal(inspection.requested)
        wait(inspection.finished)
        passed = inspection.passed
        log_inspection(id, cone, customer, passed)
        release(inspection.lock)
    signal(clerk_done)

def browse_flavours(id, cones_count):

    mess = "customer %d: asking for %d cones" % (
        id, cones_count)
    show(mess)

def walk_to_cashier(customer_id):

    mess = "customer %d: served and going to cashier" % (
        customer_id)
    show(mess)



```

```python
CLERK_COUNT = 0

def customer(id, cones_count, line, inspection):

    clerk_done = threading.Semaphore(0)

    browse_flavours(id, cones_count)

    global CLERK_COUNT
    for c in range(cones_count):
        t = threading.Thread(
            target=clerk,
            args=(CLERK_COUNT + 1,
            id, c, clerk_done, inspection))
        t.start()
        CLERK_COUNT += 1
    for c in range(cones_count):
        wait(clerk_done)

    walk_to_cashier(id)
    acquire(line.lock)
    num = line.number
    line.number += 1
    release(line.lock)
    signal(line.requested)
    wait(line.customers[num])

def check_out(i):

    mess = "cashier: customer %d paid" % i
    show(mess)

def cashier(customers_count, line):

    for i in range(customers_count):
        wait(line.requested)
        check_out(i)
        signal(line.customers[i])

def main(customers_count,
          max_cones_per_customer,
          approved_rate):

    cones = [random.choice(range(1, max_cones_per_customer))
             for i in range(customers_count)]
    tot_cones = sum(cones)
    mess = "main: %d cones %s" % (tot_cones, str(cones))
    show(mess)

    inspection = Inspection()
    line = Line(customers_count)

    cts = [threading.Thread(
        target=customer,
        args=(i,n, line, inspection))
        for i,n in enumerate(cones)]

    ct = threading.Thread(
        target=cashier,
        args=(customers_count, line))

    mt = threading.Thread(
        target=manager,
        args=(tot_cones, approved_rate, inspection))

    ct.start()
    mt.start()
    for t in cts:
        t.start()

    for t in cts:
        ct.join()
    mt.join()
    ct.join()

if __name__ == '__main__':

    import sys
    args = sys.argv[1:]
    argc = len(args)

    customers_count = 10
    max_cones_per_customer = 5
    approved_rate = 9

    if (argc > 0): customers_count = int(args[0])
    if (argc > 1): max_cones_per_customer = int(args[1])
    if (argc > 2): approved_rate = int(args[2])

    main(customers_count,
          max_cones_per_customer,
          approved_rate)
```