CS246-Assignment 10

2018/4/29

By Julia Sheng

//nothing List and node is changed, thus I did not attach them

//customer, kiosk and queue files are cited from Professor Kumar

//des.c

//simulation programs

#include <stdio.h>

#include <time.h>

#include <stdlib.h>

#include "Customer.h"

#include "Kiosk.h"

#include "Queue.h"

void simulate(int m,int k);

int arrival();

int main(int argc, char \*argv[]){

srand(time(NULL));

int m=atoi(argv[1]);

int k=atoi(argv[2]);

simulate(m,k);

return 0;

}

void simulate(int m,int k){

int queueLength=0;

Kiosk \*kiosks[k];

for(int i=0;i<k;i++)

kiosks[i]=newKiosk();

printf("There are %d kiosk.",k);

printf("[");

for(int i=0;i<k;i++)

printf("%s ",kioskToString(kiosks[i]));

printf("]\n");

Queue \*checkInQueue;

checkInQueue=newQueue();

int nc=0;

for(int t=1;t<=m;t++){

int n=arrival();

for(int i=1;i<=n;i++){

Customer \*c=newCustomer(t);

insertQ(checkInQueue,c);

nc++;

}

for(int i=0;i<k;i++){

if(!busy(kiosks[i])&&!emptyQ(checkInQueue)){

Customer \*c=removeQ(checkInQueue);

assignCustomer(kiosks[i],c);

}

tick(kiosks[i]);

}

printf("At the end of time %d there are %d customers on Q.\n",t,sizeQ(checkInQueue));

queueLength+=sizeQ(checkInQueue);

printf("[");

for(int i=0;i<k;i++){

printf("%s ",kioskToString(kiosks[i]));

}

printf("]\n");

}

printf("simulation: %d minutes, with %d kiosks\n",m,k);

printf("%d customers arrived.\n",nc);

printf("%d customers served.\n",getNServed());

float av=((float)queueLength)/(float)m;

printf("Average queue length: %.1f \n",av);

}

int arrival(){

return rand()%4;

}

// File: Customer.h

// Purpose: Defines a Customer

// Every customer has a unique ID, arrival time, and service time

// Basic functionality of using the Customer type is defined.

// Written by: Deepak Kumar

// Date: April 12, 2018

#ifndef CUSTOMER\_H

#define CUSTOMER\_H

typedef struct {

char \*id; // id is for form 'C'+# (e.g, C23, C9, etc.)

int arrivalTime;

int serviceTime;

} Customer;

Customer \* newCustomer(int time); // Given arrivale time, create a new cust

char \*getCustId(const Customer \*c); // Accessors

int getArrivalTime(const Customer \*c);

int getServiceTime(const Customer \*c);

char \*custToString(const Customer \*c); // Print form

#endif

// File: Customer.c

// Purpose: Implements a Customer defined in Customer.h

// Every customer has a unique ID, arrival time, and service time

// Basic functionality of using the Customer type is defined.

// Written by: Deepak Kumar

// Date: April 12, 2018

#include <stdio.h>

#include <stdlib.h>

#include "Customer.h"

static int count = 0; // keeps track of # customers created, which forms ID

char \*genCustId() { // creates a unique id as 'C'+count (e.g. C23, c9, etc)

count++;

char \*id = malloc(5\*sizeof(char));

sprintf(id, "C%d", count);

return id;

} // getId()

Customer \*newCustomer(int time) {

Customer \*nc = malloc(sizeof(Customer));

nc->id = genCustId();

nc->arrivalTime = time;

nc->serviceTime = 5 + rand()%4; // 5-8 min

return nc;

} // newCustomer()

char \*getCustId(const Customer \*c) {

return c->id;

} // getid()

int getArrivalTime(const Customer \*c) {

return c->arrivalTime;

} // getArrivalTime()

int getServiceTime(const Customer \*c) {

return c->serviceTime;

} // getServiceTime()

char \*custToString(const Customer \*c) {

char \*result = malloc(sizeof(char)\*25);

if (c)

sprintf(result, "<%s: %d, %d>", c->id, c->arrivalTime, c->serviceTime);

else

sprintf(result, "<CUSTOMER: NULL>");

return result;

} // custToString()

// File: Kiosk.h

// Purpose: Defines a Kiosk type

// A Kiosk has an id (K12, K6, etc.), customer being served (if any)

// time remaining for customer being served

#ifndef KIOSK\_H

#define KIOSK\_H

#include "Customer.h"

typedef struct {

char \*id;

Customer \*nowServing;

int timeRemaining;

} Kiosk;

Kiosk \*newKiosk(); // Creates a new Kiosk: K#

char \*getKioskId(const Kiosk \*k);

Customer \*getCustomer(const Kiosk \*k);

void assignCustomer(Kiosk \*k, Customer \*c);

void tick(Kiosk \*k);

int busy(const Kiosk \*k);

int getNServed(); // returns total# Customers served so far in simulation

char \*kioskToString(const Kiosk \*k);

#endif

// File: Kiosk.c

// Purpose: Implements a Kiosk type defined in Kiosk.h

// A Kiosk has an id (K12, K6, etc.), customer being served (if any)

// time remaining for customer being served

#include <stdlib.h>

#include <stdio.h>

#include "Kiosk.h"

#include "Customer.h"

static int nServed = 0;

static int count = 0;

char \*genKioskId() {

count++;

char \*id = malloc(sizeof(char)\*5);

sprintf(id, "K%d", count);

return id;

} // genKioskid()

Kiosk \*newKiosk() { // Creates a new Kiosk: K#

Kiosk \*k = malloc(sizeof(Kiosk));

k->id = genKioskId();

k->nowServing = NULL;

k->timeRemaining = 0;

return k;

} // newKiosk()

char \*getKioskId(const Kiosk \*k) {

return k->id;

} // getId()

Customer \*getCustomer(const Kiosk \*k) {

return k->nowServing;

} // getCustomer()

void assignCustomer(Kiosk \*k, Customer \*c) {

k->nowServing = c;

k->timeRemaining = getServiceTime(c);

} // assignCustomer()

void tick(Kiosk \*k) {

if (k->nowServing) {

k->timeRemaining--;

if (k->timeRemaining == 0) {

k->nowServing = NULL;

nServed++;

}

}

} // tick()

int busy(const Kiosk \*k) {

return (k->nowServing != NULL);

} // busy()

int getNServed() { // returns total# Customers served so far in simulation

return nServed;

} // getNServed()

char \*kioskToString(const Kiosk \*k) {

char \*result = malloc(30\*sizeof(char));

sprintf(result, "%s: %s", k->id, (k->nowServing ? custToString(k->nowServing): "[]"));

return result;

} // kioskToString()

// File: Queue.h

// Purpose: Defines a basic Queue type

// The queue is represented using the List type...

// Written By: Deepak Kumar

// Date: April 12, 2018

#ifndef QUEUE\_H

#define QUEUE\_H

#include "List.h"

typedef struct {

List \*L;

} Queue; // A Queue is just a linked list

// Creates a new Queue

Queue \*newQueue();

// Basic necesseties...

int sizeQ(const Queue \*q);

int emptyQ(const Queue \*q);

// Core functions for a Queue

void insertQ(Queue \*q, void \*item);

void \*removeQ(Queue \*q);

#endif

// File: Queue.c

// Prupose: Implementing a basic queue

// Data Structure used is a linked list (List)

// Front of queue is a List(0), and rear is at the end of List

//

#include <stdlib.h>

#include "List.h"

#include "Queue.h"

Queue \*newQueue() {

Queue \*q = malloc(sizeof(Queue));

q->L = newList();

} // newQueue()

int sizeQ(const Queue \*q) {

return size(q->L);

} // sizeQ()

int emptyQ(const Queue \*q) {

return empty(q->L);

} // emptyQ()

void insertQ(Queue \*q, void \*item) {

add(q->L, item); // adds at the end of queue (rear)

} // insert()

void \*removeQ(Queue \*q) {

return removeItem(q->L, 0); // removes from front of queue

} // removeQ()

//Makefile

des: des.o Customer.o List.o node.o Kiosk.o Queue.o

gcc -o des des.o Customer.o List.o node.o Kiosk.o Queue.o

des.o: des.c Customer.h Kiosk.h Queue.h

gcc -c des.c

Kiosk.o: Kiosk.h Kiosk.c Customer.h

gcc -c Kiosk.c

Customer.o: Customer.h Customer.c

gcc -c Customer.c

Queue.o: Queue.c Queue.h List.h node.h

gcc -c Queue.c

List.o: List.h List.c node.h

gcc -c List.c

node.o: node.h node.c

gcc -c node.c

clean:

rm \*.o tc tk tq

**Sample Output**

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

84 customers arrived.

78 customers served.

Average queue length: 0.4

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

87 customers arrived.

80 customers served.

Average queue length: 0.1

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

87 customers arrived.

80 customers served.

Average queue length: 0.1

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

97 customers arrived.

89 customers served.

Average queue length: 1.0

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

84 customers arrived.

75 customers served.

Average queue length: 0.1

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

89 customers arrived.

81 customers served.

Average queue length: 0.4

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

89 customers arrived.

81 customers served.

Average queue length: 0.4

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

90 customers arrived.

81 customers served.

Average queue length: 0.6

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

76 customers arrived.

68 customers served.

Average queue length: 0.2

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

76 customers arrived.

68 customers served.

Average queue length: 0.2

[jsheng@powerpuff lab13]$ ./des 60 12

There are 12 kiosk.simulation: 60 minutes, with 12 kiosks

83 customers arrived.

75 customers served.

Average queue length: 0.3

**Conclusion**

The optimum number of kiosks for the problem posed is 12. In this case, the average queue length is steadily below 1 and even the outlier is not more than 3.

**Reflection**

The main task for this assignment was to combining the small pieces into a whole program, thus I did not face many difficulties. But when I try to run the program using a loop (generating several data for given number of kiosks after each run), I found number of customer served is increasing every time which should not be the case. Then I found this error is caused by nServed(), because that nServed() is cumulative when looping(it is not reset every time). Thus I choose to run the program by hand since it is not complicated to do that in this problem. (Otherwise, we could write a new function in the Kiosk.c which reset the nServed number, so we can call this function at the end of every loop, then the data can be collected automatically)

There are 12 kiosk.[K1: [] K2: [] K3: [] K4: [] K5: [] K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 1 there are 0 customers on Q.

[K1: <C1: 1, 5> K2: <C2: 1, 7> K3: [] K4: [] K5: [] K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 2 there are 0 customers on Q.

[K1: <C1: 1, 5> K2: <C2: 1, 7> K3: [] K4: [] K5: [] K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 3 there are 0 customers on Q.

[K1: <C1: 1, 5> K2: <C2: 1, 7> K3: [] K4: [] K5: [] K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 4 there are 0 customers on Q.

[K1: <C1: 1, 5> K2: <C2: 1, 7> K3: [] K4: [] K5: [] K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 5 there are 0 customers on Q.

[K1: [] K2: <C2: 1, 7> K3: <C3: 5, 6> K4: <C4: 5, 7> K5: <C5: 5, 6> K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 6 there are 0 customers on Q.

[K1: [] K2: <C2: 1, 7> K3: <C3: 5, 6> K4: <C4: 5, 7> K5: <C5: 5, 6> K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 7 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: [] K3: <C3: 5, 6> K4: <C4: 5, 7> K5: <C5: 5, 6> K6: [] K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 8 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: <C3: 5, 6> K4: <C4: 5, 7> K5: <C5: 5, 6> K6: <C8: 8, 7> K7: [] K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 9 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: <C3: 5, 6> K4: <C4: 5, 7> K5: <C5: 5, 6> K6: <C8: 8, 7> K7: <C9: 9, 6> K8: <C10: 9, 6> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 10 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: [] K4: <C4: 5, 7> K5: [] K6: <C8: 8, 7> K7: <C9: 9, 6> K8: <C10: 9, 6> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 11 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: [] K4: [] K5: [] K6: <C8: 8, 7> K7: <C9: 9, 6> K8: <C10: 9, 6> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 12 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: <C11: 12, 7> K4: <C12: 12, 7> K5: <C13: 12, 5> K6: <C8: 8, 7> K7: <C9: 9, 6> K8: <C10: 9, 6> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 13 there are 0 customers on Q.

[K1: <C6: 7, 8> K2: <C7: 8, 7> K3: <C11: 12, 7> K4: <C12: 12, 7> K5: <C13: 12, 5> K6: <C8: 8, 7> K7: <C9: 9, 6> K8: <C10: 9, 6> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 14 there are 0 customers on Q.

[K1: [] K2: [] K3: <C11: 12, 7> K4: <C12: 12, 7> K5: <C13: 12, 5> K6: [] K7: [] K8: [] K9: <C14: 14, 5> K10: [] K11: [] K12: [] ]

At the end of time 15 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: [] K3: <C11: 12, 7> K4: <C12: 12, 7> K5: <C13: 12, 5> K6: [] K7: [] K8: [] K9: <C14: 14, 5> K10: [] K11: [] K12: [] ]

At the end of time 16 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: <C16: 16, 8> K3: <C11: 12, 7> K4: <C12: 12, 7> K5: [] K6: <C17: 16, 7> K7: [] K8: [] K9: <C14: 14, 5> K10: [] K11: [] K12: [] ]

At the end of time 17 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: <C16: 16, 8> K3: <C11: 12, 7> K4: <C12: 12, 7> K5: <C18: 17, 5> K6: <C17: 16, 7> K7: [] K8: [] K9: <C14: 14, 5> K10: [] K11: [] K12: [] ]

At the end of time 18 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: <C16: 16, 8> K3: [] K4: [] K5: <C18: 17, 5> K6: <C17: 16, 7> K7: <C19: 18, 6> K8: <C20: 18, 8> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 19 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: <C16: 16, 8> K3: <C21: 19, 5> K4: <C22: 19, 5> K5: <C18: 17, 5> K6: <C17: 16, 7> K7: <C19: 18, 6> K8: <C20: 18, 8> K9: <C23: 19, 5> K10: [] K11: [] K12: [] ]

At the end of time 20 there are 0 customers on Q.

[K1: <C15: 15, 7> K2: <C16: 16, 8> K3: <C21: 19, 5> K4: <C22: 19, 5> K5: <C18: 17, 5> K6: <C17: 16, 7> K7: <C19: 18, 6> K8: <C20: 18, 8> K9: <C23: 19, 5> K10: <C24: 20, 8> K11: [] K12: [] ]

At the end of time 21 there are 0 customers on Q.

[K1: [] K2: <C16: 16, 8> K3: <C21: 19, 5> K4: <C22: 19, 5> K5: [] K6: <C17: 16, 7> K7: <C19: 18, 6> K8: <C20: 18, 8> K9: <C23: 19, 5> K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 22 there are 0 customers on Q.

[K1: [] K2: <C16: 16, 8> K3: <C21: 19, 5> K4: <C22: 19, 5> K5: [] K6: [] K7: <C19: 18, 6> K8: <C20: 18, 8> K9: <C23: 19, 5> K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 23 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: [] K3: [] K4: [] K5: [] K6: [] K7: [] K8: <C20: 18, 8> K9: [] K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 24 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: <C28: 24, 6> K3: [] K4: [] K5: [] K6: [] K7: [] K8: <C20: 18, 8> K9: [] K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 25 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: <C28: 24, 6> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: [] K6: [] K7: [] K8: [] K9: [] K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 26 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: <C28: 24, 6> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: <C32: 26, 5> K7: <C33: 26, 8> K8: [] K9: [] K10: <C24: 20, 8> K11: <C25: 21, 8> K12: <C26: 21, 7> ]

At the end of time 27 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: <C28: 24, 6> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: <C32: 26, 5> K7: <C33: 26, 8> K8: [] K9: [] K10: [] K11: <C25: 21, 8> K12: [] ]

At the end of time 28 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: <C28: 24, 6> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: <C32: 26, 5> K7: <C33: 26, 8> K8: <C34: 28, 6> K9: <C35: 28, 5> K10: [] K11: [] K12: [] ]

At the end of time 29 there are 0 customers on Q.

[K1: <C27: 23, 8> K2: [] K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: <C32: 26, 5> K7: <C33: 26, 8> K8: <C34: 28, 6> K9: <C35: 28, 5> K10: [] K11: [] K12: [] ]

At the end of time 30 there are 0 customers on Q.

[K1: [] K2: <C36: 30, 5> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: [] K7: <C33: 26, 8> K8: <C34: 28, 6> K9: <C35: 28, 5> K10: <C37: 30, 8> K11: [] K12: [] ]

At the end of time 31 there are 0 customers on Q.

[K1: [] K2: <C36: 30, 5> K3: <C29: 25, 8> K4: <C30: 25, 8> K5: <C31: 26, 8> K6: [] K7: <C33: 26, 8> K8: <C34: 28, 6> K9: <C35: 28, 5> K10: <C37: 30, 8> K11: [] K12: [] ]

At the end of time 32 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: <C36: 30, 5> K3: [] K4: [] K5: <C31: 26, 8> K6: <C39: 32, 5> K7: <C33: 26, 8> K8: <C34: 28, 6> K9: [] K10: <C37: 30, 8> K11: <C40: 32, 8> K12: [] ]

At the end of time 33 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: <C36: 30, 5> K3: <C41: 33, 6> K4: <C42: 33, 8> K5: [] K6: <C39: 32, 5> K7: [] K8: [] K9: [] K10: <C37: 30, 8> K11: <C40: 32, 8> K12: [] ]

At the end of time 34 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: [] K3: <C41: 33, 6> K4: <C42: 33, 8> K5: <C43: 34, 5> K6: <C39: 32, 5> K7: [] K8: [] K9: [] K10: <C37: 30, 8> K11: <C40: 32, 8> K12: [] ]

At the end of time 35 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: <C44: 35, 6> K3: <C41: 33, 6> K4: <C42: 33, 8> K5: <C43: 34, 5> K6: <C39: 32, 5> K7: [] K8: [] K9: [] K10: <C37: 30, 8> K11: <C40: 32, 8> K12: [] ]

At the end of time 36 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: <C44: 35, 6> K3: <C41: 33, 6> K4: <C42: 33, 8> K5: <C43: 34, 5> K6: [] K7: <C45: 36, 5> K8: [] K9: [] K10: <C37: 30, 8> K11: <C40: 32, 8> K12: [] ]

At the end of time 37 there are 0 customers on Q.

[K1: <C38: 32, 7> K2: <C44: 35, 6> K3: <C41: 33, 6> K4: <C42: 33, 8> K5: <C43: 34, 5> K6: <C46: 37, 8> K7: <C45: 36, 5> K8: <C47: 37, 7> K9: [] K10: [] K11: <C40: 32, 8> K12: [] ]

At the end of time 38 there are 0 customers on Q.

[K1: [] K2: <C44: 35, 6> K3: [] K4: <C42: 33, 8> K5: [] K6: <C46: 37, 8> K7: <C45: 36, 5> K8: <C47: 37, 7> K9: <C48: 38, 7> K10: [] K11: <C40: 32, 8> K12: [] ]

At the end of time 39 there are 0 customers on Q.

[K1: <C49: 39, 6> K2: <C44: 35, 6> K3: [] K4: <C42: 33, 8> K5: [] K6: <C46: 37, 8> K7: <C45: 36, 5> K8: <C47: 37, 7> K9: <C48: 38, 7> K10: [] K11: [] K12: [] ]

At the end of time 40 there are 0 customers on Q.

[K1: <C49: 39, 6> K2: [] K3: <C50: 40, 5> K4: [] K5: <C51: 40, 6> K6: <C46: 37, 8> K7: [] K8: <C47: 37, 7> K9: <C48: 38, 7> K10: <C52: 40, 6> K11: [] K12: [] ]

At the end of time 41 there are 0 customers on Q.

[K1: <C49: 39, 6> K2: <C53: 41, 6> K3: <C50: 40, 5> K4: <C54: 41, 7> K5: <C51: 40, 6> K6: <C46: 37, 8> K7: [] K8: <C47: 37, 7> K9: <C48: 38, 7> K10: <C52: 40, 6> K11: [] K12: [] ]

At the end of time 42 there are 0 customers on Q.

[K1: <C49: 39, 6> K2: <C53: 41, 6> K3: <C50: 40, 5> K4: <C54: 41, 7> K5: <C51: 40, 6> K6: <C46: 37, 8> K7: [] K8: <C47: 37, 7> K9: <C48: 38, 7> K10: <C52: 40, 6> K11: [] K12: [] ]

At the end of time 43 there are 0 customers on Q.

[K1: <C49: 39, 6> K2: <C53: 41, 6> K3: <C50: 40, 5> K4: <C54: 41, 7> K5: <C51: 40, 6> K6: <C46: 37, 8> K7: <C55: 43, 6> K8: [] K9: <C48: 38, 7> K10: <C52: 40, 6> K11: [] K12: [] ]

At the end of time 44 there are 0 customers on Q.

[K1: [] K2: <C53: 41, 6> K3: [] K4: <C54: 41, 7> K5: <C51: 40, 6> K6: [] K7: <C55: 43, 6> K8: [] K9: [] K10: <C52: 40, 6> K11: [] K12: [] ]

At the end of time 45 there are 0 customers on Q.

[K1: [] K2: <C53: 41, 6> K3: [] K4: <C54: 41, 7> K5: [] K6: [] K7: <C55: 43, 6> K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 46 there are 0 customers on Q.

[K1: <C56: 46, 5> K2: [] K3: <C57: 46, 5> K4: <C54: 41, 7> K5: <C58: 46, 7> K6: [] K7: <C55: 43, 6> K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 47 there are 0 customers on Q.

[K1: <C56: 46, 5> K2: <C59: 47, 8> K3: <C57: 46, 5> K4: [] K5: <C58: 46, 7> K6: [] K7: <C55: 43, 6> K8: [] K9: [] K10: [] K11: [] K12: [] ]

At the end of time 48 there are 0 customers on Q.

[K1: <C56: 46, 5> K2: <C59: 47, 8> K3: <C57: 46, 5> K4: <C60: 48, 7> K5: <C58: 46, 7> K6: <C61: 48, 5> K7: [] K8: <C62: 48, 5> K9: [] K10: [] K11: [] K12: [] ]

At the end of time 49 there are 0 customers on Q.

[K1: <C56: 46, 5> K2: <C59: 47, 8> K3: <C57: 46, 5> K4: <C60: 48, 7> K5: <C58: 46, 7> K6: <C61: 48, 5> K7: <C63: 49, 5> K8: <C62: 48, 5> K9: <C64: 49, 8> K10: <C65: 49, 7> K11: [] K12: [] ]

At the end of time 50 there are 0 customers on Q.

[K1: [] K2: <C59: 47, 8> K3: [] K4: <C60: 48, 7> K5: <C58: 46, 7> K6: <C61: 48, 5> K7: <C63: 49, 5> K8: <C62: 48, 5> K9: <C64: 49, 8> K10: <C65: 49, 7> K11: <C66: 50, 5> K12: <C67: 50, 6> ]

At the end of time 51 there are 0 customers on Q.

[K1: [] K2: <C59: 47, 8> K3: [] K4: <C60: 48, 7> K5: <C58: 46, 7> K6: <C61: 48, 5> K7: <C63: 49, 5> K8: <C62: 48, 5> K9: <C64: 49, 8> K10: <C65: 49, 7> K11: <C66: 50, 5> K12: <C67: 50, 6> ]

At the end of time 52 there are 0 customers on Q.

[K1: <C68: 52, 5> K2: <C59: 47, 8> K3: [] K4: <C60: 48, 7> K5: [] K6: [] K7: <C63: 49, 5> K8: [] K9: <C64: 49, 8> K10: <C65: 49, 7> K11: <C66: 50, 5> K12: <C67: 50, 6> ]

At the end of time 53 there are 0 customers on Q.

[K1: <C68: 52, 5> K2: <C59: 47, 8> K3: [] K4: <C60: 48, 7> K5: [] K6: [] K7: [] K8: [] K9: <C64: 49, 8> K10: <C65: 49, 7> K11: <C66: 50, 5> K12: <C67: 50, 6> ]

At the end of time 54 there are 0 customers on Q.

[K1: <C68: 52, 5> K2: [] K3: <C69: 54, 6> K4: [] K5: <C70: 54, 7> K6: <C71: 54, 5> K7: [] K8: [] K9: <C64: 49, 8> K10: <C65: 49, 7> K11: [] K12: <C67: 50, 6> ]

At the end of time 55 there are 0 customers on Q.

[K1: <C68: 52, 5> K2: <C72: 55, 8> K3: <C69: 54, 6> K4: <C73: 55, 7> K5: <C70: 54, 7> K6: <C71: 54, 5> K7: <C74: 55, 5> K8: [] K9: <C64: 49, 8> K10: [] K11: [] K12: [] ]

At the end of time 56 there are 0 customers on Q.

[K1: [] K2: <C72: 55, 8> K3: <C69: 54, 6> K4: <C73: 55, 7> K5: <C70: 54, 7> K6: <C71: 54, 5> K7: <C74: 55, 5> K8: <C75: 56, 7> K9: [] K10: <C76: 56, 5> K11: <C77: 56, 5> K12: [] ]

At the end of time 57 there are 0 customers on Q.

[K1: <C78: 57, 7> K2: <C72: 55, 8> K3: <C69: 54, 6> K4: <C73: 55, 7> K5: <C70: 54, 7> K6: <C71: 54, 5> K7: <C74: 55, 5> K8: <C75: 56, 7> K9: <C79: 57, 6> K10: <C76: 56, 5> K11: <C77: 56, 5> K12: [] ]

At the end of time 58 there are 0 customers on Q.

[K1: <C78: 57, 7> K2: <C72: 55, 8> K3: <C69: 54, 6> K4: <C73: 55, 7> K5: <C70: 54, 7> K6: [] K7: <C74: 55, 5> K8: <C75: 56, 7> K9: <C79: 57, 6> K10: <C76: 56, 5> K11: <C77: 56, 5> K12: <C80: 58, 6> ]

At the end of time 59 there are 2 customers on Q.

[K1: <C78: 57, 7> K2: <C72: 55, 8> K3: [] K4: <C73: 55, 7> K5: <C70: 54, 7> K6: <C81: 59, 6> K7: [] K8: <C75: 56, 7> K9: <C79: 57, 6> K10: <C76: 56, 5> K11: <C77: 56, 5> K12: <C80: 58, 6> ]

At the end of time 60 there are 1 customers on Q.

[K1: <C78: 57, 7> K2: <C72: 55, 8> K3: <C82: 59, 6> K4: <C73: 55, 7> K5: [] K6: <C81: 59, 6> K7: <C83: 59, 8> K8: <C75: 56, 7> K9: <C79: 57, 6> K10: [] K11: [] K12: <C80: 58, 6> ]

simulation: 60 minutes, with 12 kiosks

84 customers arrived.

74 customers served.

Average queue length: 0.1