Cairo University

First year Computer Engineering Department

Data Structures and Algorithms Project

Name	Sec	BN
Osama Magdy Aied	1	14
Radwa Ahmed Mohamed	1	30
Nada Elsayed Mohamed	2	31
Yousef Ahmed Anwar	2	37

Cairo University, Faculty of Engineering Computer Engineering Department Data Structures and Algorithms CMP102/CMPN102	
Data Structures and Algorithms **Final Assessment Report**	
Team Name: T6	**Number of members : 4**
Email : nadaelsayed163@gmail.com **2132000@gmail.com** **yousef.elftah00@eng-st.cu.edu.eg** **osamamagdy174@gmail.com**	

Section1: Osama Magdy Aied, 114

1. Function _ Name

Checkunavailblecooks()

1.1. Member of

Restaurant

1.2. Inputs

timestep

1.3. Returns

Void(no return)

1.4. Called by

Restaurnat:: Restaurant_modes()

1.5. Calls

///////////////All Queue Functions for different data types ///////////////////////////

I. Queue::isEmpty()
II. Queue::enqueue()
III. Queue::dequeue()

// They are used for different Queues: BusyCooks, BreakCooks, InjuredCooks, Ncooks, Gcooks, Vcooks, Prepare_Order

///////////////////////////////////////Cook Functions /////////////////////////////

IV. Cook::getRstPrd()
V. Cook::getN_orders_Finished()
VI. Cook::get_order_to_break())
VII. Cook::setServedOrder()
VIII. Cook::setN_orders_Finished()
IX. Cook::getBreakduration()
X. Cook::setAvailableNcount()
XI. Cook::GetAvailableNcount()
XII. Cook::setAvailableGcount()
XIII. Cook::GetAvailableGcount()
XIV. Cook::setAvailableVcount()
XV. Cook::GetAvailableVcount()
XVI. Cook::getOriginalSpeed()
XVII. Cook::GetType()
XVIII. Cook::getInjProp()
XIX. Cook::GetStatus()
XX. Cook::getServedOrder())
XXI. Cook::setStatus()
XXII. Cook::setSpeed()
XXIII. Cook::getspeed()
XXIV. Cook::increase_injury()

XXV. Cook::setTimesteptobeavailabale()
XXVI. Cook::getTimesteptobeavailabale()
XXVII. Cook::CalUnavailabalePriority()
XXVIII. Cook::getUnavailabalePriority()

//Order Functions///////////////////////////

XXIX. Order::setservicetime()
XXX. Order::getservicetime()
XXXI. Order::CalFinish();
XXXII. Order::getFinshtime()

1.6. Logic Description

This function is responsible for updating all non-available cooks whether to be busy with orders, having a break or getting rest from injuries and also generates injuries for cooks randomly.

a) For BusyCooks, we pick the first one to see if it's the time for him to be available and dequeue it. If not, we don't do anything, setting our flag to false and go for the next check as our BusyCooks are stored in priority queue depending on the TimeStep when they became available(The least TimeStep has the highest priority)

   After that we dequeue it from BusyCooks as this means he finished his order and increment the number of orders he finished. If he was injured during working on this order we make him go to rest after we make sure that his finished orders are less than orders to get a break (if not, we set the number of orders to 0 and consider his break to be taken during his rest period). Then, we set the timestep to be available equals to current + restperiod and enqueue it to InjuredCooks Queue.

   If he wasn't injured, we check for the number of orders he made to see if he deserves a break or not and enqueue him to the BreakCooks queue if he reaches the number of orders to have a break with the priority of the timestep to be available. Last thing if all previous conditions were false we make him return to the kitchen as an available cook depending on his type to be assigned to new order in the next tinestep.

b) For BreakCooks, we check if the first Cook has finished his break to dequeue him and return to available queue depending on his type

c) For InjuredCooks, we check if the first Cook has finished his restperiod to dequeue him and return to available queue depending on his type

d) To generate injured cooks we take a random value and if it’s less than the injury probability in the load files we pick the first non-injured cook from BusyCooks queue and his order from prepare order queue, then we reduce his speed to the half, set his status to injured and make the time step to be available increase by the double of the time remaining to finish. We do the same for his order by increasing its service time and the finish time as well. At the end we enqueue them again to their queues with new priorities

2. Function _ Name

assignOrderInjured()

2.1. Member of

Restaurant

2.2. Inputs

Timestep, Order, msg

2.3. Returns

Bool

2.4. Called by

Restaurnat:: urgentForVIP( )

2.5. Calls

Queue::isEmpty(), Queue::enqueuer(), Queue::dequeuer(),Order::setStatus(),Order::setwaittime(),Order:: setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook:: setServedOrder(),

Order::setTimesteptobeavailabale(),Order:: setN_orders_Finished(),Order:: CalUnavailabalePriority()

2.6. Logic Description

This function is called only to assign orders for injured cooks when there is an Urgent VIP Order that waited too much and there is no available cooks or cooks in break, this happens by picking the first cook in InjuredCooks queue and assigning him this order to be served. We notice that the order will take double the time to be served as the Cook's speed is down to half. We set the status of the cook to be BUSY and enqueue him in BusyCooks queue with the new timestep to be available. It also adds a message that the order has been assigned to that cook and says both Type and ID for the cook and Order. After the Cook finishes his order he returns to available cooks queue as we changed his status to BUSY so checkunavailable() function won't return him back to InjuedCooks queue but his speed will still the half of his original speed until he takes a break. We enqueue the Order to the PrepareOrder Queue with priority depending on his finish time.

3. Function _ Name

assignOrderBreak

3.1. Member of

Restaurant

3.2. Inputs

Timestep, Order, msg

3.3. Returns

Bool

3.4. Called by

Restaurnat:: urgentForVIP()

3.5. Calls

Queue::isEmpty(), Queue::enqueuer(),

Queue::dequeue(),Order::setStatus(),Order::setwaittime(),Order::

setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook::

setServedOrder(),

Order::setTimesteptobeavailabale(), Order:: setN_orders_Finished(),

Order:: CalUnavailabalePriority()

3.6. Logic Description

This function is called only to assign orders for Break cooks when there is an Urgent VIP Order that waited too much and there is no available cooks or cooks, this happens by picking the first cook in BreakCooks queue and assigning him this order to be served. We set the status of the cook to be BUSY and enqueue him in BusyCooks queue with the new timestep to be available. It also adds a message that the order has been assigned to that cook and says both Type and

ID for the cook and Order. After the Cook finishes his order he returns to available cooks queue as we changed his status to BUSY so checkunavailable() function won't return him back to BreakCooks queue until he takes another break. We enqueue the Order to the PrepareOrder Queue with priority depending on his finish time.

4. Function _ Name

shellSort

4.1. Member of

Restaurant

4.2. Inputs

Order* arr[], int n,

4.3. Returns

Void(no return)

4.4. Called by

Restaurnat:: assignOrdertofinish

4.5. Calls

getservicetime()

4.6. Logic Description

ShellSort is mainly another type of Insertion Sort Algorithms. We start by making a gap between elements and compare elements of the array that have this gap in between. Then we start to reduce the gap to the half after we loop through all elements until we get the array to be sorted.

Section2: Radwa Ahmed, 130

5. Function _ Name:

AddOrders

5.1. Member of:

Restaurant

5.2. Inputs:

Order

5.3. Returns

Void(no return)

5.4. Called by

Event::ArrivalEvent(), Restaurant::autopormotedForNormal()

5.5. Calls

Enqueue functions of each type of orders and calls functions type of each order and if the order is VIP, the function calls priority function.

Queue::enqueuer(), Queue :: Dequeue(), PQueue::enqueuer(), PQueue::Dequeue()

Order::getPrioirity(), Order::GetType()

5.6. Logic Description

Add each order in queues. It adds the orders in three different queues: VIP, Vegan, Normal.

6. Function _ Name

autopormotedForNormal

6.1. Member of

Restaurant

6.2. Inputs

Timestep

6.3. Returns

Void(no return)

6.4. Called by

Restaurnat:: Restaurant_modes

6.5. Calls

Queue::isEmpty(), Queue::peekFront() Queue::dequeue(),Order:: getorderarrivaltime(), Order:: GetAUto(),Order:: SetType(),Restaurant::AddOrders() , Order::setNOrderscount(),

Order:: increase_promotion(),Order:: setVOrderscount(),Order:: set_time_when_became_VIP(time);

6.6. Logic Description

It checks if the queue of Normal order is not empty, then peek the order and check if his timestep – arrival time becomes equals to time of auto. If the condition is true ,it will dequeuer the order and change the type to VIP and increase number of VIP orders and decrease the number of normal order and add the order to VIP queue.

7. Function _ Name

urgentForVIP

7.1. Member of

Restaurant

7.2. Inputs

Timestep, string msg

7.3. Returns

Void(no return)

7.4. Called by

Restaurnat:: Restaurant_modes

7.5. Calls

Order:: increase_urgent(),

Queue::isEmpty(), Queue::enqueuer(), Queue::dequeuer(),Order::setStatus(),Order::setwaittime(),Order:: setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook:: setServedOrder(),Restauarnt:: assignOrderInjured(),Restaurnat::assginOrder Break(),

Order:: setTimesteptobeavailabale(),Order:: setN_orders_Finished(),Order:: CalUnavailabalePriority()

7.6. Logic Description

This function check if the urgent order is not empty it try to assign it to VIP cook ,if all are busy, it try to assign to normal then to vegan cooks. If they are busy, it will try to assign it to break cook by calling assginbreak then injured cook by calling assigninjured. If all are not found, it will try in the next time step. If it assigned it will dequeue the cook and the order. It will enqueue the cook in busy cook and the order will be enqueued in prepare orders. It will then enqueue the cook using priority and after changing the status and the number of serverd orders and time to be available. It changes the order to be servces and set the waiting time and change the status of the order to be SRV and enqueue in prepare.

8. Function _ Name

CheckurgentForVIP

8.1. Member of

Restaurant

8.2. Inputs

Timestep

8.3. Returns

Void (no return)

8.4. Called by

Restaurnat:: Restaurant_mode

8.5. Calls

Queue::peekFront, Queue ::queue(), Queue ::dequeuer(),Order::get_time_when

_became_VIP(),Order::getVIP_WT().

8.6. Logic Description

If the orders becomes urgent, it will dequeue it from VIP order and enqueue it in urgent vip.

9. Function _ Name

Excute

9.1. Member of

Promotion event

9.2. Inputs

Timestep,Id,extraMoney

9.3. Returns

Void(no return)

9.4. Called by

Excute event

9.5. Calls

Restaurant::Search,Order::SetType, Order::AddMoney, Restaurant::Addorders, Order::setNOrderscount, Order::setVOrderscount, Order::set_time_when_became_VIP

9.6. Logic Description

It checks if extra money is larger than zero, then dequeuer the normal order from its queue using search function. It changes the type to become VIP, and the time of Vip. It also change number of vip orders and number of normal orders. After that add the vip queue.

Section3: Nada Elsayed, 231

10. Function _ Name

Search

10.1. Member of

Restaurant

10.2. Inputs

Order by reference, ID.

10.3. Returns

Void (no return)

10.4. Called by

Event::Cancellationevent(), Event::promotionevent()

10.5. Calls

Queue::Enqueue() , Queue::Dequeue() , Order::GetID()

10.6. Logic Description

It takes a normal order ID and search in normal orders queue then return this order by reference. It is using in searching for cancel a normal order.

11. Function _ Name

assignOrderVIP

11.1. Member of

Restaurant

11.2. Inputs

Timestep, string msg

11.3. Returns

Bool

11.4. Called by

Restaurnat:: Restaurant_modes

11.5. Calls

Queue::isEmpty(), Queue::enqueuer(), Queue::dequeuer(),Order::setStatus(),Order::setwaittime(),Order:: setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook:: setServedOrder(),

Order:: setTimesteptobeavailabale(),Order:: setN_orders_Finished(),Order:: CalUnavailabalePriority()

11.6. Logic Description

It is assigning the VIP order to cook depends on the assigning criteria that check VIP cooks first then Normal cooks then finally vegan cooks. If it found an available cook then,

- It dequeue the cook from the available queue (Vcooks, Ncooks or Gcooks) then it enqueus this cook in busy queue after change his statue to busy, set this order as a served order, set the time that this cook will finish this order depending on the cook's speed and number of order dishes, and finally it decreases available cooks' counter minus one and store this order ID and cook ID as a string message.
- It also dequeue the order from VIP orders queue then it enqueue the order to prepare queue after change his statue to serviced, set wait time to the current time step minus the arrival time step, set service time by divide the number of dishes on the cook speed, it set the time that this order will be finished in, and finally it increases VIP orders counter plus one.

12. Function _ Name

assignOrderVegan

12.1. Member of

Restaurant

12.2. Inputs

Timestep , string msg

12.3. Returns

Bool

12.4. Called by

Restaurnat:: Restaurant_modes

12.5. Calls

Queue::isEmpty(), Queue::enqueuer(),

Queue::dequeuer(),Order::setStatus(),Order::setwaittime(),Order:: setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook:: setServedOrder(),

Order:: setTimesteptobeavailabale(),Order:: setN_orders_Finished(),Order:: CalUnavailabalePriority()

12.6. Logic Description

It is assigning the vegan order to cook depends on the assigning criteria that vegan cook only who can make this order. If it found an available cook then,

- It dequeue the cook from the available queue (Gcooks) then it enqueus this cook in busy queue after change his statue to busy, set this order as a served order, set the time that this cook will finish this order depending on the cook's speed and number of order dishes, and finally it decreases available cooks' counter minus one and store this order ID and cook ID as a string message.
- It also dequeue the order from vegan orders queue then it enqueue the order to prepare queue after change his statue to serviced, set wait time to the current time step minus the arrival

time step, set service time by divide the number of dishes on the cook speed, it set the time that this order will be finished in, and finally it increases vegan orders counter plus one.

13. Function _ Name

assignOrderNormal

13.1. Member of

Restaurant

13.2. Inputs

Timestep ,string msg

13.3. Returns

Bool

13.4. Called by

Restaurnat:: Restaurant_modes

13.5. Calls

Queue::isEmpty(), Queue::enqueuer(), Queue::dequeuer(),Order::setStatus(),Order::setwaittime(),Order:: setservicetime(),Order:: CalFinish(),Cook:: setStatus(), Cook:: setServedOrder(),

Order::setTimesteptobeavailabale(),Order:: setN_orders_Finished(),Order:: CalUnavailabalePriority()

13.6. Logic Description

It is assigning the Normal order to cook depends on the assigning criteria that check Normal cooks first then VIP cooks. If it found an available cook then,

- It dequeue the cook from the available queue (Ncooks or Vcooks) then it enqueus this cook in busy queue after change his statue to busy, set this order as a served order, set the time that this cook will finish this order depending on the cook's speed and number of order dishes, and finally it decreases available cooks' counter minus one and store this order ID and cook ID as a string message.
- It also dequeue the order from normal orders queue then it enqueue the order to prepare queue after change his statue to serviced, set wait time to the current time step minus the arrival time step, set service time by divide the number of dishes on the cook speed, it set the time that this order will be finished in, and finally it increases normal orders counter plus one.

Section4: Yousif Ahmed, 237

14. Function _ Name

loadfile

14.1. Member of

Restaurant

14.2. Inputs

Void (no input)

14.3. Returns

Void (no return)

14.4. Called by

Restaurnat::RunSimulation()

14.5. Calls

-GUI::GetString().

-Cook::setVcount().

-Cook::setNcount().

-Cook::setGcount().

-Cook::setordertobreak().

-Order::setautopormotion().

-Order::SetVIP_WT().

-Cook::SetRstPrd().

-Cook::SetInjProp().

-Queue::enqueue().

-Cook::setSpeed().

-Cook::setOriginalSpeed().

-Cooks::setType().

-Cook::setBreakduration().

-Cook::setID().

14.6. Logic Description

In this function we get file name from user then read this file line by line and set data in its data member then read events line by line and enqueue it with different status (arrival, cancellation and promotion) and different type.

15. Function _ Name

Outputfile()

15.1. Member of

Restaurant

15.2. Inputs

Void (no input)

15.3. Returns

Void (no return)

15.4. Called by

Restaurnat:: RunSimulation()

15.5. Calls

-Queue::toArray()

-Restaurnat::getavgSTandWT().

-GUI::PrintMessage().

- GUI::GetString().

-Order::getordercount().

-Order::getNordercount().

-Order::getGordercount().

-Order::getVordercount().

-Cook::Getcookscount().

-Cook::GetNcount().

-Cook::GetGcount().

-Cook::GetVcount().

-Cook::get_num_of_injury().

-Order::Get_num_of_order_auto_P().

-Order::get_Urgent_num().

-Order::getFinshtime().

-Order::GetID().

-Order::getorderarrivaltime().

-Order::getwaittime().

-Order::getservicetime().

15.6. Logic Description

In this function we take output file name from user then writing output line by line,
after sorting finish order using shell sort algorithm and getting pointer to it, we print its data line by line in each line we print finish time, id, arrival time, wait time and service time.
we get order counter and the counter of each type, and cooks counter and counter of each type.
We print the average of service time and waiting time of finish orders, then we get urgent order number then auto promoted orders number.

16. Function _ Name

getavgSTandWT( )

16.1. Member of

Restaurant.

16.2. Inputs

- arr (array of pointer to Order).

- count (number of Order in the array).

- avgWT (float data sending by reference to get average of Waiting time).

- avgST (float data sending by reference to get average of Service time).

16.3. Returns

Void (no return).

16.4. Called by

Restaurant:: Outputfile().

16.5. Calls

-Order::getwaittime().

-Order::getservicetime().

16.6. Logic Description

In this function we calculate the average of Waiting time and the average of Service time**.**

We calculate the sum of waiting and service time of each order finish,

then we divide it by finish orders number.

17. Function _ Name

assignOrdertofinish

17.1. Member of

Restaurant.

17.2. Inputs

Timestep.

17.3. Returns

Void (no return).

17.4. Called by

Restaurnat::Restaurant_modes().

17.5. Calls

-Queue::isEmpty(),

-Queue::peekFront(),

-Queue::dequeue(),

-Order::setStatus().

-Order:: getFinshtime().

-Order:: setFinishedOrdersCount().

-Restaurant::shellSort().

-Queue::enqueue().

17.6. Logic Description

-In each time step we call this function and updates every order that has been finished during this timestep.

Frist, we get the peek front of the prepareOrder queue to see if the first order has been finished by comparing the order finish time and the current time step and if it finishes in this time step we dequeue it from prepareOrder and increase finish order count then change order status to DONE and we repeat this operation until all there is no order in prepareOrder is finish in this time step, if not, we return from the function.

Then we store the finished orders in an array and sort this array depending on their service time (as they all have the same finish time) using shell sorting algorithm. After that we enqueue elements of the sorted array to the finished order queue and deletes the array after that.

18. Function _ Name

Restaurant_modes()

18.1. Member of

Restaurant

18.2. Inputs

Mode (int number for each mode)

18.3. Returns

Void (no return)

18.4. Called by

Restaurnat::RunSimulation().

18.5. Calls

-GUI::PrintMessage().

-GUI::waitForClick().

-Queue::isEmpty().

-Restaurant::ExecuteEvents().

-Restaurant::assignOrdertofinish().

-Restaurant::checkunavilblecooks().

-Restaurant::assignOrderVIP().

-Restaurant::assignOrderVegan().

-Restaurant::assignOrderNormal().

-Restaurant::urgentForVIP().

-Restaurant::autopormotedForNormal().

-Restaurant:: FillDrawingList().

18.6. Logic Description

In this function we do all modes as we get selected mode from the

user and receive it working in each mode as defined.
-We start our time step then we call Execute Events function with current time step to get all events which happen in this time step.

- Then we start to assign Orders according to Orders Criteria:

1- assign VIP-Orders until waiting VIP is empty or all cooks is not available.

2-If there is no VIP-Waiting Orders assign Vegan-Orders until Vegan-waiting order is empty or all Vegan-Cooks is not available.

3-Then we assign Normal Order until Normal-Waiting Order is empty or all cook is unavailable.

-Then we call urgentForVIP() to check if there is an urgent VIP-order.
-Then we call autopormotionForNormal() to check if there is a normal Order need to auto promote.
-All above process happen in each mode
Interactive mode, Step by step mode and Silence mode.
-For Interactive mode each time step we wait for user click to advance current time step then draw data using FillDrawingList().
-For Step-by-Step mode we wait for one second then we draw data using FillDrawingList() then we advance current time step.