|  |  |
| --- | --- |
| Cairo University, Faculty of Engineering  Computer Engineering Department  Data Structures and Algorithms  CMP102/CMPN102 | Spring 2020 |

**Data Structures and Algorithms**

**Final Assessment Report**

|  |  |
| --- | --- |
| **Team Name:** | **Number of members:4** |
| **Email:** |  |

**Section1: Zeyad Ossama Mostafa,1180034**

**Function 1: MoveToAvailable**

**Member** **of**: Restaurant

**Inputs**:

N : 1

X: pointer of Type cook

**Returns**:

**Called** **By**:

* Restaurant:: MovingBreakToAvailable
* Restaurant:: MovingRestToAvailable
* Restaurant:: CheckBusyCooks()

**Calls**:

**Function** **Logic** **description**:

It checks the cook type then moves it to the corresponding Available cook list

**Function2:** **MovingBreakToAvailable**

**Member** **of**: Restaurant

**Inputs**:

N : 0

X:

**Returns**:

**Called** **By**:

* Restaurant:: assignmentfunction

**Calls**:

* Resturant:: MoveToAvailable

**Function** **Logic** **description**:

It Check the cooks In break cooks List if the current timestep equals the timestep in which the cook will be available the pass the cook to MoveToavailable function which move it to the corresponding available cook list to its type,and also checks if the cook Status is Urgent and ended his break then reset it status to safe and multiply it's speed by 2

**Function3:** **MovingRestToAvailable**

**Member** **of**: Restaurant

**Inputs**:

N : 0

X:

**Returns**:

**Called** **By**:

* Restaurant:: assignmentfunction

**Calls**:

* Resturant:: MoveToAvailable

**Function** **Logic** **description**:

It Check the cooks In rest cooks List if the current timestep equals the timestep in which the cook will be available it pass the cook to MoveToavailable function which move it to the corresponding available cook list to its type,and also reset it status to safe and multiply it's speed by 2

**Function4:** **CheckBusyCooks**

**Member** **of**: restaurant

**Inputs**:

N : 0

X:

**Returns**:

Returns the ………

**Called** **By**:

* Restaurant:: assignmentfunction

**Calls**:

* Restaurant:: MoveToAvailable
* Restaurant:: SRV\_to\_Finshed

**Function** **Logic** **description**:

It creates random number between 0 to 1 if the number is smaller or equal injury probability then makes the first cook in Busycooks list injured and decrease it speeds to half and sets its status as injured then dequeues it from the list and reset its availability time and enqueue it again as the busy the busycooks list is prorityqueue the order of the injured cook will change,then checks the cooks in busycooks list if the availability time (the time in which the cook will be available) is equal the current timestep it checks if the number of orders served by the cook equal the number of orders before break then moved to in breakcooks list,if not then checks if he is injured If the cook injured then moved to In rest cooks list else it passed to moveToAvailable Function

**Function5:** **SRV\_to\_Finshed**

**Member** **of**: Restaurant

**Inputs**:

N : 1

X: pointer of type order

**Returns**:

**Called** **By**:

* Restaurant:: CheckBusyCooks

**Calls**:

**Function** **Logic** **description**:

It gets the ID if the passed order the loops on the in serving Orders and checks the ID of every order in serving list if the ID is the same then it removed from the list

**Section2: Amr Mohamed, 1180472**

**Function 1: WaitingOrdersToServed**

**Member** **of**: Class Restaurant

**Inputs**:

No inputs

**Returns**:

Doesn’t return

**Called** **By**:

* Restaurant::assignmentfunction()

**Calls**:

* WaitingOrderVIPenqueue(Order\*)
* WaitingOrderVIPdequeuePriority()
* WaitingOrderVIPdequeue()

**Function** **Logic** **description**:

In the current timestep, the function checks if there are any urgent orders. If so, the function searches for the most adequate cook for this order to serve including cooks in break or rest. After it finishes all the urgent orders, it checks for vip cook and assign them to the most adequate cook. After that, the same happens for the Vegan orders and Normal orders. At the assignment process the order is removed from its waiting queue and added to the InServing queue. The cooks are removed from their queue and added to the Busycook priority queue.

**Function 2: WaitingOrderVIPenqueue**

**Member** **of**: Class Restaurant

**Inputs**:

Order\*

**Returns**:

Doesn’t return

**Called** **By**:

* Restaurant:: WaitingOrdersToServed()
* Restaurant::promotion(int, double)

**Calls**:

* None

**Function** **Logic** **description**:

The function enqueue the order taken from the input to the priority queue **VIP\_OrdersWaitingPriorityQueue** and the normal queue **VIP\_OrdersWaiting**.

**Function 3: WaitingOrderVIPdequeue**

**Member** **of**: Class Restaurant

**Inputs**:

None

**Returns**:

Order\*

**Called** **By**:

* Restaurant:: WaitingOrdersToServed()

**Calls**:

* None

**Function** **Logic** **description**:

It dequeues an order form the Norma queue **VIP\_OrdersWaiting** then searches in the priority queue **VIP\_OrdersWaitingPriorityQueue** for that order then dequeues it and return the order dequeued from the 2 queues.

**Function 4: WaitingOrderVIPdequeuePriority**

**Member** **of**: Class Restaurant

**Inputs**:

None

**Returns**:

Order\*

**Called** **By**:

* Restaurant:: WaitingOrdersToServed()

**Calls**:

* None

**Function** **Logic** **description**:

It dequeues an order form the priority queue **VIP\_OrdersWaitingPriorityQueue** then searches in the normal queue **VIP\_OrdersWaiting** for that order then dequeues it and return the order dequeued from the 2 queues.

Section 3: Abdulrehman Mahmoud Suliman, 1180140

**Function 1: promotion**

**Member of:** Class Restaurant

**Inputs:**

N:2

int: Id of the order to promote

double: Extra money the customer paid to promote his order.

**Returns:**

None

**Called By:**

Restaurant::promotion(int , double)

**Calls:**

Order::GetID()

Order::SetType(ORD\_TYPE)

Order::SetMoney(double)

**Function Logic description:**

This functions gets two inputs ,the id of the order to promote and the extra money the customer paid to promote his order so this function searches in the normal waiting orders queue for the order with the given id and when it is found the order is removed from the normal waiting queue and put in the VIP waiting queue and increment the orders money with the given amount. This process happens as the function dequeues the orders from the normal waiting queue and checks if the order’s id equals to the given id if yes it enqueues the order in the VIP waiting queue and complete dequeuing and inqueuing the orders in the normal waiting queue until the first order returns to its place again and by the this the function checked all the orders.

**Function 2: Execute**

**Member of:** PromotionEvent

**Inputs:**

N: 1

X : Pointer from Restaurant

**Returns:**

None

**Called By:**

PromotionEvent::Execute(Restaurant\*)

**Calls:**

Restaurant::promotion(int,double)

**Function logic description:**

this function executes the promotion event by calling the promotion function from Restaurant Class.

**Function 3:OutputFunction**

**Member of:** Class Restaurant

**Inputs:**

None

**Returns:**

None

**Called by:**

Restaurant::OutputFunction()

**Calls:**

Order::getWaitingTime()

Order::getServingTime()

Order::getID()

Order::getArrivalTime()

ofstream::open(“OutPutFile”)

ofstream::close()

**Function logic description:**

First in this function we get the average waiting time and the average waiting time by dequeuing the orders from the finished orders queue and incrementing a variables with the values of waiting and serving times and then equeue the order again in the same queue and this process continues until the first order in the queue returns to it’s original place which means that the function checked all the orders in the queue then calculate the two averages it also calculates the percentage of the autopromoted orders then the orders are out in a priority queue which sorts the orders ascending according to the finish time then generates an output file which prints all the orders info the number of orders and cooks and number of each type of them and gives the count of the injured cooks as well and the urgent orders and the autopromoted orders percentage.

**TestCase\_1**

**TestCase\_2**

**TestCase\_3**

**TestCase\_4**

**TestCase\_5**

**TestCase\_6**