|  |  |  |
| --- | --- | --- |
| **Credit Hours System**  **CMPN102-Spring2020** |  | **Cairo University**  **Faculty of Engineering** |

Final Assessment: Restaurant simulation

**Submitted to :** ENG.Eman, ENG.Abdelrahman Eed

**Students name :** Ahmad Mostafa El Sayed (1180018)

: Karim Amr Mohamed (1180118)

: Mohamed Amr Mohamed(1180010)

: Abdallah Ahmad Hassan (1180011)

**Team** : 2-3707, 8-11 AM

|  |  |  |  |
| --- | --- | --- | --- |
| |  |  | | --- | --- | | 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: 2 3707** | **Number of members: 4** |
| **Email: karimamr9@outlook.com** |  |
| **Section1: Ahmad Mostafa El Sayed, 1180018**  **Function1\_assignToCook:**  **Member** **of**: Restaurant  **Inputs**:  Order\*:a pointer to the order about to be assigned  **Returns**:  This function return a Boolean value that is true if the order actually got assigned to a cook and is false if none of the cooks are free  **Called** **By**:   * Restaurant::assigningOrders() * Restaurant::AssignUrgentOrder()   **Calls**:   * Cook::AssignOrder(orderToAssigned,currentTimeStep) * Also the standard operations on queues(deqeue,enqeue,peek)   **Function** **Logic** **description**:  This function is responsible for handling a single order and assigning it to a cook depending on its type , This function is a Boolean by choice because we needed this return value to be used in the assign urgent order function , So basically this function takes the order checks its type and match it to a cook from the available cooks queue and removes the cook from his respective queue and removes the order too , and enqueues the cook to the nuys list and the order to the in service list.  **Function1\_moveFromInservToFinished:**  **Member** **of**: Restaurant  **Inputs**:  No inputs  **Returns**:  Void, does not return anything  **Called** **By**:   * Restaurant::InterActiveMode( ) * Restaurant::Step\_by\_StepMode( ) * Restaurant::SilentMode( )   **Calls**:   * Cook::removeOrder() * Also the standard operations on queues(deqeue,enqeue,peek)   **Function** **Logic** **description**:  This function is responsible for moving all finished orders from the in service list to the  Finished orders list and returning back the cooks to their respective lists depending on their type or whether they need a break or whether they need to go to rest after injury , this function checks all the orders queues and checks if this is the timestep that that the order is supposed to be finished in or not , and it also increments the total wait and total serv time times for calculating outputs in the save file.  **Function3\_assigingOrders:**  **Member** **of**: Class restaurant  **Inputs**:  No inputs  **Returns**:  Void, does not return anything  **Called** **By**:   * Restaurant::InterActiveMode( ) * Restaurant::Step\_by\_StepMode( ) * Restaurant::SilentMode( )   **Calls**:   * Restaurant:assignToCook()   **Function** **Logic** **description**:  This function is responsible only for calling the assignToCook function every time step  In all orders that are still in waiting and if the order is assigned it removes it from its list. | |
| **Section2: Member Name: Karim Amr Mohamed Mohamed**  **Function1\_saveToFile:**  **Member of** :Restaurant  **Inputs:** none  **Returns :** return type is void  **Called by:**   * Restaurant::InterActiveMode( ) * Restaurant::Step\_by\_StepMode( ) * Restaurant::SilentMode( )   **Calls**: doesn’t call any major functions  **Function logic description:**  This function gets called in the end of every mode first it makes sure that finished orders are sorted by finish time first then arrival time, then opens a save file (saveFile.txt) and outputs the required numbers in the format mentioned in the document and calculating the averages.    **Function2\_** **checkEndBreakOrRest:**  **Member of:** Restaurant  **Inputs:** none  **Returns**: return type is void  **Called by:**   * Restaurant::InterActiveMode( ) * Restaurant::Step\_by\_StepMode( ) * Restaurant::SilentMode( )   **Calls**: Cook::returnToAction(int t) and standard operations queues(dequeue,enqueue,peek)  **Function logic description:**  As on break cooks and on rest cooks are each placed in a priority queue according to the cook with the shortest break/rest we check the first entry in both queues if returnToAction() returns true then this cook finished his break/rest and returned to the appropriate queue and incrementing the available cooks counter and we check the next cook in the break/rest queues.  **Function3\_** **returnToAction:**  **Member of:** Cook  **Inputs:** int timeStep  **Returns**: bool  **Called by:**   * Restaurant:: checkEndBreakOrRest()   **Calls**: None  **Function logic description:**  Function takes current time step and returns true if the cook finished his break or rest  And if the cook was injured, it removes injury flag by setting it to false and restores his original speed. Otherwise it returns false. | |