Priority Delivery

**1)Testing menu option 1: Display Inventory, function void displayInventoryDetails(pQueue pQueueObj).**

**Description:** Function will display the number of widgets available in the inventory and also the price Per widget. Function calculates quantity of widgets available by subtracting the total Widget stock with the number of widgets ordered (by calling pQueueObj.getCountWidgetsOrdered()). There are enough widgets to complete all orders.

**Error**: Function not displaying correct data.

b)Testing function with no widgets ordered.

**Expected output:** Function will Display the total amount of Widgets available for the day. Also, the price per widget.

a)Testing function after orders were received.

**Expected output:** Function will display the number of widgets available, by subtracting from the total number of widgets the number of widgets ordered throughout the day.

**2)Testing menu option 2 functions:** **void receiveOrderMenu(pQueue &pQueueObj), void Enquque(int numwidgetsOrdered, RushStatus rushStatus).**

**A)function void receiveOrderMenu(pQueue &pQueueObj)**

**Description:** Function will ask the user for number of widgets he/she would like to order, if the input is correct the rush status menu will be printed to the screen, after the user enters an option from 1 to 3, the ordered is added in the priority Queue, with the respective priority for the order.

**Error:** Wrong user input.

a)Testing function with correct user input:

**Expected output:** Order is going to be added to the priority Queue, respecting the priority as needed.

b)Testing function with wrong user input:

**Expected Output:** Function will display the appropriate message and ask user to renter an order until user enter correct input.

**B)function Enquque(int numwidgetsOrdered, RushStatus rushStatus).**

**Description:** Function takes the user input and ads it to the priority queue, which is implemented as a heap. After element is added function ReheapUp() is called to reorganize the heap array so that the heap priority is restored.

**Error:** Priority Queue is full ,or priority is not respected.

a)Testing function with valid user input

**Expected output:** Order will be added to the Queue and afterwards by reheping the order will be added at the correct index so that it respects the heap priority. Order will be printed to the screen with its tracing number, to confirm that the order was received and added to the Queue.

b)testing function with full Queue

**Expected output:** Error message explaining that the Queue is full will be displayed on the screen.

**3)Testing menu option #3 functions: void closeDay(pQueue &pQueueObj) ;void PrintOrders(pQueue& pQueueObj); void Dequeue(Order& orderWidget)**

**A)Function void closeDay(pQueue &pQueueObj);**

**Description:** Function is going to display the details of all the orders processed, after closing the day, along with details like total orders processed, total cost to warehouse, total profit, total cost to customer and inventory on hand. Function also calls function **PrintOrders(pQueue& pQueueObj);** to print all the orders that were processed that day, in the order they were shipped.

**Error:** pQueue is empty**.**

**a)**Testing function with pQueue not empty, with orders available.

**Expected output**: Function will display the total order details, plus all the orders processed that day with there details, in the order they were shipped.

b)Testing function with no orders available (pQueue empty).

**Expected output:** Error message explaining user that the order queue is empty will be printed to the screen.

**B) Function PrintOrders(pQueue& pQueueObj);**

**Description:** Function will display all the orders processed in a day in the order they were shipped, by calling Dequeue function and coping the top element from the Queue before removing it and using the respective object to print details.

Testing function: Same process as function: **void closeDay(pQueue &pQueueObj);**

**C)Function void Dequeue(Order& orderWidget)**

**Description:** Function removes elementwith the highest priority in pQueue (Root of the heap) and return a copy of that element. After removing the root, void ReheapUp(int root, int bottom) is called to restore the heap property.

**Error:** pQueue is empty

a)Testing function with pQueue not empty:

**Expected output:** Function will remove the element with the highest priority and return a copy of the element. Functions calls reheapDown function to restore the heap property.

b)Testing function with qQueue empty:

**Expected output:** Function is going to print an error message explaining that he pQueue is empty.

**4)Testing function void ReheapDown(int root, int bottom); And function void ReheapUp(int root, int bottom);**

**A)** **Function** **ReheapDown(int root, int bottom);**

**Description:** Function restores the order property after the root element is removed.

**Error:** Not restoring the order correctly

a)Testing function by removing element ( order)

**Expected output:** Heap order property is restored.

**B)Function void ReheapUp(int root, int bottom);**

**Description:** Function changes the position of a newly added element to the heap Array, by swapping until the heap order property is restored.

**Error:** Not restoring the order correctly

1. Testing function by adding new orders to the pQueue:

**Expected output:** Function will swap the element with it’s parent until heap property is restored **.**

**5)Testing Calculation functions: double calculateOrderPrice(Order tepmOrder);**

**double calculateTotalCostToCustomer();**

1. **Function: double calculateTotalCostToCustomer()**

**Description:** Functioncalculates the total cost for all customers, respecting the rush status Markup for each individual order, and returns the total.

**Error:** Function does not calculate properly, or no available orders;

a)Testing function with multiple orders;

**Expected output**: Function will return the total sum of all the customer orders price, respecting the markup percentage regarding rush status.

b)Testing function with no available orders:

**Expected output**: Function will return 0.

**B)Function**: **calculateOrderPrice(Order tepmOrder);**

**Description:** Function will return the cost for one order.

**Error:** Not calculating properly, and not respecting markup percentage

a)Testing function with an extreme rush order:

**Expected output:** Function will return the number of widgets ordered times the widget price time the markup percentage which is 100% for extreme rush.

b)Testing function with an expedite rush order:

**Expected output:** Function will return the number of widgets ordered times the widget price times the markup percentage which is 40% for expedite rush.’

b)Testing function with and standard rush order:

**Expected output:** Function will return the number of widgets ordered times the widget price times the markup percentage which is 20% for standard rush.