Test Plan Backward Warehouse

1. **Testing menu option 1.Details of inventory on hand: Function** void displayinventoryStackDetails():

**-Specification:** If the inventory stack is empty appropriate message will be printed, else all available batches and their details(batch number, quantity on hand and price per widget) will be printed to the screen.

**Error** is true if inventory stack is empty.

A)**Test plan for void displayinventoryStackDetails():**

a) Testing function with an empty inventory stack:

**Expected Output:** Error message explaining that the inventory stack is empty will be displayed on to the screen.

b) Testing function with the inventory stack having available batches for orders:

**Expected Output:** all batches in the inventory stack will be displayed on the screen, and their details(batch number, quantity on hand and price per widget).

1. **Testing menu option 2.Details of outstanding deliveries: Function** void displayOutstandingOrders():

**-Specification:** If order stack is empty appropriate message will be printed, else all available orders present in the order stack and their details(order number, quantity to be shipped) will be printed to the screen.

**Error** is true if order stack is empty.

**A)Test Plan for void displayOutstandingOrders():**

a)Testing with empty order stack:

**Expected Output:** Error messageexplaining that the order stack is empty will be displayed on to the screen.

b)Testing function with orders available in the order stack:

**Expected Output:** all orders and their details ( order number, quantity to be shipped) will be displayed on the screen.

**3.Testing menu option 3.Accept a delivery:** Functions acceptDelivery(InventoryStack& inventoryStackObj, OrderStack& orderStackObj)

void processOrder(InventoryStack& inventoryStackObj, OrderStack& orderStackObj, float widgetPrice, int numWidgetsOrdered, float& costToWarhouse)

**Specification:** Ask user for the number of widgets and the price per widget for the delivery, if valid input new delivery will be accepted and pushed onto the inventory stack. After the delivery is accepted program will check if there are any outstanding orders in the order stack, if there exists any unprocessed order’s function processOrder() will be called and any unprocessed orders will be processed first.

**Error:** Wrong user input

**A)Test Plan for acceptDelivery(InventoryStack& inventoryStackObj, OrderStack& orderStackObj):**

a)Testing function with wrong user input:

**Expected output:** Error message will be printed to the screen explaining error and user will be able to renter the information for the order.

b)Testing function with correct user input:

**Expected output:** If the Inventory Stack is not full the, all the details regarding the batch quantity and the price per widget will be pushed onto the inventory stack, and batch tracking number will be assigned, and the number of widgets will be added to countNumWidgetsInStock(counter for the number of widgets in stock) . If the inventory stack is full Error message will be printed to the screen and delivery will not be processed/pushed onto the stack.

**B)Test Plan for void processOrder(InventoryStack& inventoryStackObj, OrderStack& orderStackObj, float widgetPrice, int numWidgetsOrdered, float& costToWarhouse)**

a)Testing function in case Accept delivery:

**Expected output:** If order stack has orders available to be shipped, function processOrder(…) is called and information from the new delivery that was accepted will be processed by the function. If enough widgets were received to fulfill all orders. All orders will be processed and shipped, countNumWidgetsShipped (counter that keeps track of the number of widgets shipped) will be increased by the number of widgets that were shipped. Details regarding each order that was fulfilled will be printed to the screen. If not, enough widgets were received, all orders that can be fulfilled will be fulfilled, and orders that can not be fully fulfilled will have a part of the order shipped and the rest will be shipped ass soon as new deliveries are received by the warehouse. Total Cost to the warehouse for each fulfilled or partially fulfilled will be calculated, with the prices from each individual batch in the inventory.

**4.Testing menu option 4.Take an order, Functions:**

void takeAnOrder(InventoryStack& obj, OrderStack& orderObj)

void processOrder(InventoryStack& inventoryStackObj, OrderStack& orderStackObj, float widgetPrice, int numWidgetsOrdered, float& costToWarhouse)

**Specification:** Function will ask user for the number of widgets he would like to order, if input is correct the order is going to be pushed onto the order stack, if there are any available widgets in stock, widgets will be sent to processes by calling function processOrder(). Order will be either fully, partially, or not at all completed. Depending on the number of widgets available in inventory. If enough widgets are in stock to fulfill the order, order will be shipped, if not enough widgets available, a part of the order will be shipped or not at all(depending on if there are any widgets available) and a backorder message will be printed to the screen, letting user now that his order will be processed as soon as inventory will receive more widgets.

**Error** wrong user input.

**A)Test Plan for function: void takeAnOrder(InventoryStack& obj, OrderStack& orderObj)**

a)Testing function with wrong user input:

**Expected output**: Error message will be printed to the screen and user will be able to renter the number of orders he would like to order until the input is valid .

b)Testing function with correct user input:

**Expected output:** If order stack is not full,Order will be pushed onto the order stack and inventory will be checked for available widgets if widgets are in stock, the widgets will be shipped to fulfill or partly fulfill an order, countNumWidgetsShipped (counter that keeps track of the number of widgets shipped) will be increased by the number of widgets that were shipped.

**5. Testing function void processOrder(InventoryStack& inventoryStackObj, OrderStack& orderStackObj, float widgetPrice, int numWidgetsOrdered, float& costToWarhouse) as an individual.**

**Specifications:** Function will process any unfulfilled orders and return the total cost to the Warehouse for each individual order that was fully or partly fulfilled.

**Errors:** Function not processing orders properly, and not calculating the cost to the warehouse properly.

1. **Testing function with widgets in stock to fulfill or partly fulfill order:**

**Expected Output:** Function will process orders at the top of the stack first, if order is fulfilled order is shipped, details regarding order will be printed to the screen and function will check if inventory still has widgets in stock and if there are any orders in the order stack, if both are valid function continues to process orders until no more widgets in stock or no more orders to process. cost to warehouse will be calculated with each widget that is sent to fulfill an order and each individual price per widget.

**B)Testing function with an order bigger than number of widgets in stock**

**Expected Output:** Function will send all available widgets to partly fulfill the order, and a backorder message will be printed along with the details regarding the backorder.

**C)Testing function with orders and no available widgets**

**Expected Output:** Function will print a message letting the user now that no items are available to be sent for order, and that they will receive the items ass soon as the inventory receives a delivery of widgets.

**D) Testing function with orders pending and inventory receiving stock**

**Expected Output:** All orders starting from the top of the stack will be shipped until all orders are full or partly fulfilled, or until inventory stack has no more widgets available in stock.

**6.Testing option menu Function:** void OptionMenu(int& optionNum)

**Specification:** Function will print the option menu for the user to interact with it and return the option number inputted by the user.

**Error:** Wrong input from user (Number greater than number of options or input string, char ,negative number)

**A)Test plan for function void OptionMenu(int& optionNum)**

**a)**Testing with correct input

**Expected output**: Function will return the option number; option will be executed, and option menu will be printed again.

**b)**Testing function with incorrect input:

**Expected output**: Function will display an error message; option menu will be reprinted to the screen and user will be able to renter an option from the menu.