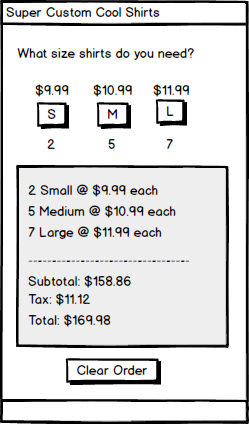
### **90%**

### **Hands-On Test Grading Rubric**

This test assesses your ability to apply concepts from your textbook, labs, and lecture notes. Please ensure all submitted code is your original work. Your solution must be uploaded to GitHub in the /HandsOnTest/CH6/ folder with the solution name HOT6P2 for grading. No partial credit will be given.

#### **EX1: ShirtSizes (50 points)**



**Objective:** Create a program for ordering T-Shirts where users can select sizes, add them to an order, clear the order, and view an invoice. Shirt prices are fixed, and sales tax is 7%.

**Rubric Breakdown:**

* **User Interface Layout (5 points):**
  + **Clarity:** All UI elements (buttons for sizes, order display, "Clear Order" button, invoice display) are clearly arranged and visually align with the provided screen mockups.
* **Tab Order Configuration (5 points):**
  + **Accessibility:** Users can navigate through all interactive UI elements (e.g., buttons) sequentially and logically using the Tab key.
* **Control Naming Conventions (5 points):**
  + **Consistency:** All UI controls in your code (e.g., btnSmallShirt, lblTotalPrice, btnClearOrder) follow a consistent, descriptive naming convention (e.g., Hungarian notation, PascalCase, or camelCase, as per your course's standard).
* **Variable Naming Conventions (5 points):**
  + **Readability:** All variables used throughout your code (e.g., shirtPrice, orderTotal, taxRate) are named descriptively and follow standard naming conventions, making their purpose clear.
* **AddShirt(size) Method Implementation (7 points):**
  + **Method Header:** AddShirt(string size)
  + **Accuracy:** This method correctly processes the size of the shirt (e.g., "Small", "Medium") passed to it.
  + **Order Management:** It accurately adds the selected shirt to the user's order and updates the displayed list of ordered items.
  + **UI Update :** Dynamically updates subtotals as items are added
* **ClearOrder() Method Implementation (8 points):**
  + **Method Header:** ClearOrder()
  + **Functionality:** This method successfully removes *all* items from the current order.
  + **Reset State:** It correctly clears the displayed order list and resets all associated financial calculations (subtotal, tax, total) to zero or their initial values.
* **ShowInvoice() Method Implementation (15 points):**
  + **Method Header:** ShowInvoice()
  + **Calculation Accuracy:** This method correctly calculates the subtotal of all ordered shirts, applies the 7% sales tax, and determines the accurate grand total.
  + **Display:** It clearly presents the final invoice details to the user, including the itemized list, subtotal, tax amount, and grand total.

#### 

#### 

#### 

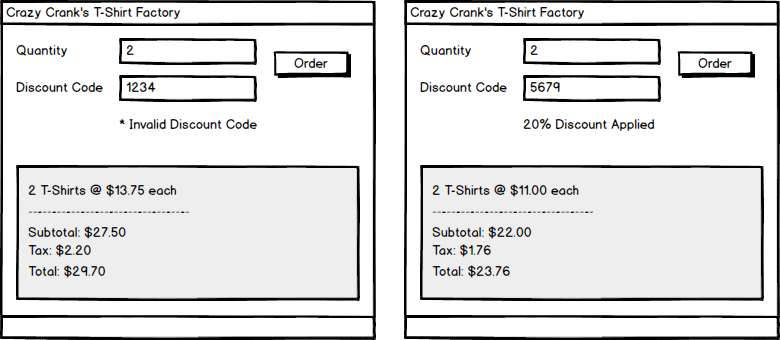
#### 

#### 

#### 

#### 

#### **EX2: ShirtDiscounts (50 points)**



**Objective:** Create a program for ordering T-Shirts where users enter a quantity, an optional discount code, and view an invoice. Shirts are $13.75 each, sales tax is 8%, and discounts apply only if a valid code is entered.

**Discount Codes:**

* 8264: 30% off
* 5679: 20% off
* 6483: 10% off

**Rubric Breakdown:**

* **User Interface Layout (5 points):**
  + **Clarity:** All UI elements (quantity input, discount code input, "Apply Discount" or "Show Invoice" button, invoice display) are clearly arranged and visually align with the provided screen mockups.
  + **Intuitiveness:** The layout is logical and easy for a user to understand and interact with.
* **Tab Order Configuration (5 points):**
  + **Accessibility:** Users can navigate through all interactive UI elements (e.g., input fields, buttons) sequentially and logically using the Tab key.
* **Control Naming Conventions (5 points):**
  + **Consistency:** All UI controls in your code (e.g., txtQuantity, txtDiscountCode, btnShowInvoice) follow a consistent, descriptive naming convention.
* **Variable Naming Conventions (5 points):**
  + **Readability:** All variables used throughout your code (e.g., itemQuantity, discountPercentage, totalBeforeTax) are named descriptively and follow standard naming conventions.
* **CheckDiscountCode(code) Method Implementation (10 points):**
  + **Method Header:** CheckDiscountCode(string code) : float
  + **Input Handling:** This method accurately receives the discount code string as an input.
  + **Return Value:** It correctly returns the corresponding discount percentage as a float (e.g., 0.30 for 30%) if the code is valid. If the code is invalid or empty, it returns 0.0 (or another appropriate value indicating no discount).
* **Discount Code Validation Logic (10 points):**
  + **Loop Usage:** The CheckDiscountCode() method explicitly uses an iterative control structure (e.g., a for loop, while loop, or similar iteration technique) to search through a collection of valid discount codes.
  + **Accuracy:** It correctly matches the entered code against the predefined valid codes to determine the accurate discount percentage.
* **ShowInvoice(quantity, discountPercentage) Method Implementation (10 points):**
  + **Method Header:** ShowInvoice(int quantity, float discountPercentage)
  + **Parameter Usage:** This method correctly accepts the quantity of shirts and the discountPercentage (from CheckDiscountCode()) as inputs.
  + **Calculation Accuracy:** It accurately calculates the cost based on quantity, applies the discountPercentage if greater than zero, calculates the subtotal, adds the 8% sales tax, and determines the final grand total.
  + **Display:** It clearly presents the final invoice details to the user, including quantity, original price, applied discount (if any), subtotal, tax amount, and grand total.