87% - Nice Work!

## **C# Hands-On Test Rubric: Detailed Instructions**

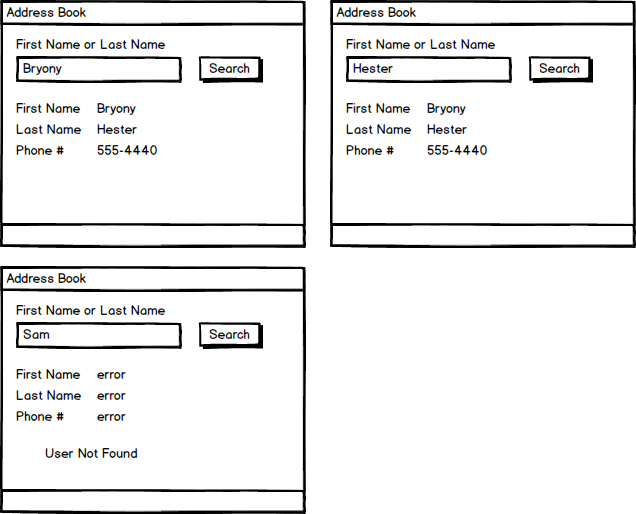
This document outlines the requirements and grading criteria for your C# Hands-On Test. Please utilize your textbook, lab exercises, and online resources to complete the test. All submitted code must be your original work.

**Submission Requirements:**

* Place both projects within a single solution.
* The solution and project files must be located inside the /HandsOnTest/CH08/ folder structure. **(5 points)**

### **Project 1: C# WinForms GUI Application - Contact Information Lookup (55 points)**

**Objective:** Develop a C# WinForms application that allows users to search for contact information by first or last name.



**Data Storage:**

* The application must store the following contact data using **three parallel arrays or List<string> objects**:
  + First Name | Last Name | Phone Number
  + Markel | Diggory | 555-8390
  + Luiza | Gunnar | 555-4618
  + Bryony | Hester | 555-4440
  + Giraldo | Addy | 555-1687
  + Lowri | Hari | 555-7763

**Functionality:**

1. **Search Input:** Provide a user interface element (e.g., TextBox) for the user to enter a search query.
2. **Search Logic:**
   * Implement a search mechanism that allows users to search for a contact by their first name **or** their last name.
   * The search must be **case-insensitive** (e.g., "markel" should match "Markel").
   * The search must support **partial matches** (e.g., "Mark" should find "Markel").
   * The search operation must be performed using a **single loop** that iterates through your data structures.
3. **Display Results:**
   * **Successful Search:** If a contact is found, display their complete information (first name, last name, and phone number) in appropriate UI elements (e.g., Label controls).
   * **Failed Search:** If no contact matches the search criteria, display all expected error messages clearly to the user.

**Technical Requirements:**

* **Control Naming:** All WinForms controls (e.g., buttons, text boxes, labels) must adhere to standard C# naming conventions (e.g., btnSearch, txtFirstName, lblPhoneNumber).
* **Variable Naming:** All variables within your code must follow standard C# naming conventions (e.g., firstName, lastNameList, phoneNumberArray).
* **Loop Usage:** Utilize an appropriate loop construct for searching through the contact data.

**Grading Criteria for Project 1:**

* **5 pts:** Controls names follow C# naming conventions.
* **3 pts:** Variable names follow C# naming conventions.

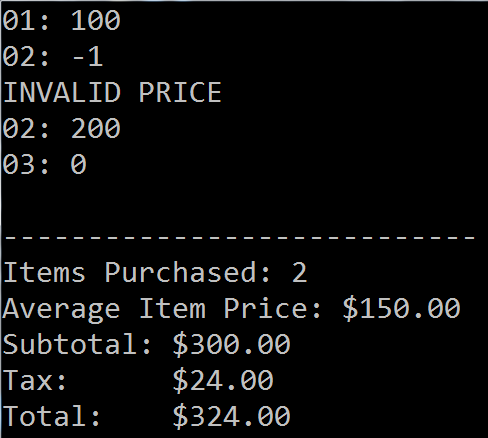


* **5 pts:** Program correctly uses the arrays or lists to store, search, and display the data.
* **5 pts:** Program uses a single loop to search for users.
* **5 pts:** Failed searches display all expected error messages.
* **4 pts:** User can successfully search for a user by their first name.
* **4 pts:** User can successfully search for a user by their last name.
* **4 pts:** User can perform a case-insensitive search.
* **5 pts:** User can perform partial searches (e.g., searching for "Mark" finds "Markel").
* A screenshot of a computer

  AI-generated content may be incorrect.
* **5 pts:** Program accurately displays the first name of the found user.
* **5 pts:** Program accurately displays the last name of the found user.
* **5 pts:** Program accurately displays the phone number of the found user.

### **Project 2: C# Console Application - Shopping Cart Calculator (40 points)**

**Objective:** Create a C# console application that calculates the total cost of items in a shopping cart, including tax, based on user input.



**Functionality:**

1. **Price Input Loop:**
   * The program must continuously prompt the user to enter the price of an item until the user enters 0 (zero).
   * **Valid Price (Greater than 0):**
     + Accept the entered price.
     + Prompt the user for the next item's price.
   * **Invalid Price (Less than 0):**
     + Ignore the entered price.
     + Display a clear error message to the user (e.g., "Invalid price. Please enter a positive number or zero to finish.").
     + Prompt the user for the next item's price.
   * **Termination Price (Zero):**
     + Ignore the entered price (do not add it to the total).
     + Do not display an error message.
     + Proceed to display the summary calculations.
     + Stop the program execution.
2. **Calculations and Output:**
   * Once the user enters 0, the program must calculate and display the following information:
     + The total number of items purchased.
     + The average price per item.
     + The subtotal (sum of all valid item prices before tax).
     + The total tax (calculated at an 8% rate of the subtotal).
     + The grand total (subtotal + total tax).

**Technical Requirements:**

* Utilize appropriate loop(s) to handle the continuous input of prices.

**Grading Criteria for Project 2:**

* **5 pts:** Program correctly accepts and processes valid prices (>0).
* **5 pts:** Program correctly identifies and rejects invalid prices (<0), displaying an error message.
* **5 pts:** Program stops accepting input and proceeds to calculations only when 0 is entered.
* **5 pts:** Program accurately displays the total number of items purchased.
* **5 pts:** Program accurately displays the average price per item.
* **5 pts:** Program accurately displays the subtotal (before tax).
* **5 pts:** Program accurately displays the total tax (8% of subtotal).
* A screen shot of a black screen

  AI-generated content may be incorrect.
* **5 pts:** Program accurately displays the grand total (subtotal + tax).