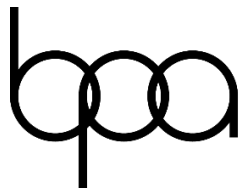


Contestant ID: _____

Time: _____

Rank: _____



**BUSINESS
PROFESSIONALS
of AMERICA**
Giving Purpose to Potential

C# PROGRAMMING

(330)

REGIONAL 2026

PRODUCTION:

Regional_C_Sharp

_____ (1030 points)

Test Time: 90 minutes

GENERAL GUIDELINES.

Failure to follow any of these rules may result in disqualification:

1. **Submission Requirements:** Contestants must submit this test booklet along with any printouts.
2. **Permitted Items:** Only the equipment, supplies, and materials specified for this event are allowed in the testing area. Previous BPA tests and sample tests (whether handwritten, photocopied, or typed) are not permitted.
3. **Electronic Devices:** Electronic devices will be monitored according to ACT standards.

EXAM GUIDELINES.

1. Your name and/or school name should *not* appear on work you submit for grading.
2. Create a folder on the flash drive provided using your Contestant ID as the name of the folder.
3. Copy your entire solution/project into this folder. The project folder for you has already been provided: Regional_C_Sharp
4. Submit your entire solution/project so that the graders may open your project to review the source code.
5. Ensure that the files required to run your program are present and will execute on the flash drive provided.
6. You will need to use Visual Studios 2019 or greater to complete this exam.

*Note that the flash drive letter may *not* be the same when the program is graded as it was when you created the program.

The graders will *not* alter your source code. Submissions that do *not* contain source code will *not* be graded.

C# Programming Overview

Welcome to the Regional C# programming competition! This assessment is designed to evaluate your understanding of core programming concepts in C# and your ability to implement functionality within a Windows Forms application. You will work with a pre-built application, adding logic and features to several buttons to bring the program to life.

What You'll Be Doing

1. Implement the Logic for Various Buttons:

- Implement the logic for **12 buttons** in the application. Each button has a specific task, ranging from simple string manipulations to more complex operations like financial calculations and dynamic UI updates.

2. Simulate and Test Operations:

- Ensure the program runs correctly and handles user inputs effectively, including displaying appropriate error messages for invalid inputs.

3. Add Comments:

- Add comments to specific parts of your code as outlined to demonstrate understanding of the programming concepts involved.

Grading Criteria:

- Each button implementation is assigned a specific point value based on its complexity. To score well:
 - Ensure your implementation meets all requirements for the button.
 - Use clear and efficient code.
 - Include thoughtful comments where required.
 - And make sure your program is working when you submit it so you can still earn points; you can still win without programming all of the buttons.

Commenting for Source Code Review:

- Certain sections of your code will be graded. These gradable blocks of code can range from creating data structures, method algorithms, exception handling, and class construction.
 - Code Commenting Requirements: Clear and concise comments must be included for all major components of the program, specifically:
 - Input: Document how data is received or acquired, including user inputs, file reads, the format of the input, API requests, etc.
 - Processing: Provide comments that explain the logic, algorithms, or transformations applied to the input data.
 - Output: Describe how and where the final results are produced, the format of the output, whether through user interfaces, files, or other systems.
 - These comments should enhance understanding of the program's flow and intent, making it easier for others to read, maintain, and debug the code.
- The grading rubric contains a section called Source Code Review: in this section are listed descriptions of all the graded programming concepts.
- Each gradable item and any significant programming area should be commented. There will be points awarded for proper commenting.

Objects Contained in Form1:

| | |
|---|---|
| <p>Text Boxes (txt):</p> <ul style="list-style-type: none"> • txt_Prin • txt_Interest • txt_Years • txt_Backwards • txt_NoVowels • txt_AddEvens • txt_Digits • txt_Distance <p>List Boxes (lst):</p> <ul style="list-style-type: none"> • lst_VocabGen • lst_MonthlyPayment • lst_Distance <p>Buttons (btn):</p> <p>btn_VocabGen btn_MonthlyPayment btn_Backwards btn_NoVowels btn_AddEvens btn_Digits btn_Distance btn_GraderButton btn_RandomFont btn_Add btn_Subtract btn_HideShow</p> | <p>Radio Buttons (rdb):</p> <ul style="list-style-type: none"> • rdb_Miles • rdb_KM <p>Labels (lbl):</p> <ul style="list-style-type: none"> • lbl_Principal • lbl_Int • lbl_Years • lbl_Backwards • lbl_NoVowels • lbl_RandomFont • lbl_AddSubtract • lbl_Convert |
|---|---|

Windows Form1 Template Design

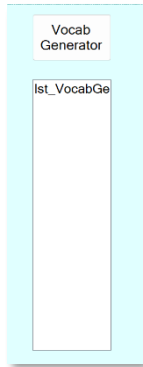
See the image below to familiarize yourself with the template of the Windows Form you will be programming. You will not be required to adjust any of the objects in the form, and all of the programmable event method signatures have been created for you (i.e. clicking on a button).

The screenshot displays a Windows Form titled "Form1" with a standard Windows title bar (minimize, maximize, close buttons). The form is divided into several colored panels, each containing specific controls and labels:

- Light Blue Panel (Top Left):** Contains a "Vocab Generator" label and a text box labeled "lst_VocabGe".
- Light Green Panel (Top Middle):** Contains a "Monthly Payment" label, a "Principal" text box, an "Interest Paid" text box, a "Time in Years" text box, and a "lst_MonthlyPayment" text box.
- Light Orange Panel (Top Right):** Contains a "Backwards" label, an "Enter Here" text box, a label "lbl_Backwards", a "No Vowels" label, another "Enter Here" text box, and a label "lbl_NoVowels".
- Red Panel (Top Far Right):** Contains an "Add Evens" label, an "Enter Here" text box, a "Digit Types" label, and another "Enter Here" text box.
- Light Blue Panel (Bottom Middle):** Contains a "Distance Converter" label, a text box, a "Convert To:" label with radio buttons for "Miles" and "Kilometers", and a text box labeled "lst_Distance".
- Light Orange Panel (Bottom Left):** Contains a "Random Font" label, the text "Random Font!!!!", and two buttons labeled "Add" and "Subtract".
- Light Orange Panel (Bottom Middle):** Contains a label "Add or Subtrct".
- Light Orange Panel (Bottom Right):** Contains a button labeled "Hide or Show".
- Light Gray Panel (Top Far Right):** Contains a button labeled "Grader Button".

Regional C Sharp: The Buttons You Will Program:

1. Vocab Generator



Button Text: "Vocab Generator"

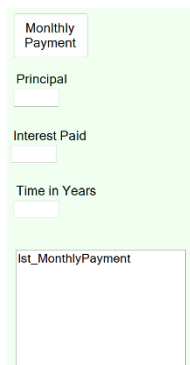
Instructions:

- Create a `List<string>` or equivalent data structure to store vocabulary terms like "Algorithm," "Binary," "Compiler," and so on.
- When the button is clicked:
 - Check if the associated `ListBox` is empty.
 - If it is empty, populate the `ListBox` with the vocabulary terms from the data structure.
 - If it is not empty, clear the `ListBox`.
 - The vocabulary words have already been provided for you.

private void btn_VocabGen_Click(object sender, EventArgs e)

- Handles the "Vocab Generator" button click event; toggles between populating and clearing a vocabulary list.

2. Monthly Payment



Button Text: "Monthly Payment"

Instructions:

- Take inputs from text boxes for:
 - **Principal** (Loan amount)
 - **Annual Interest Rate**
 - **Loan Term in Years**
- Convert the annual interest rate to a monthly rate and the loan term into months.

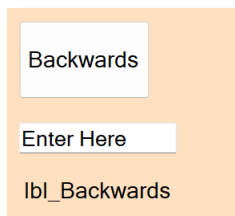
$$\text{Monthly Payment} = \frac{r \times P \times (1 + r)^n}{(1 + r)^n - 1}$$

- Use the provided financial formula to calculate the monthly payment:
- Where:
 - r = Monthly Interest Rate
 - P = Principal
 - n = Total Number of Payments
- Display the calculated monthly payment in the associated `ListBox`.
- Handle input errors by showing a message box if inputs are invalid.

private void btn_MonthlyPayment_Click(object sender, EventArgs e)

- Handles the "Monthly Payment" button click event; calculates and displays monthly payment based on loan details.

3. Backwards



Backwards

Enter Here

lbl_Backwards

Button Text: "Backwards"

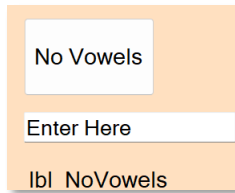
Instructions:

- Retrieve the text input from the associated `TextBox`.
- Reverse the text.
- Display the reversed text in the associated `Label`.

private void btn_Backwards_Click(object sender, EventArgs e)

- Handles the "Backwards" button click event; reverses the input text and displays it.

4. No Vowels



Button Text: "No Vowels"

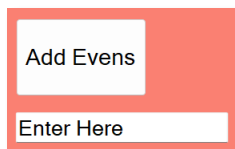
Instructions:

- Retrieve the text input from the associated `TextBox`.
- Remove all vowels (a, e, i, o, u, in both uppercase and lowercase) from the text.
- Display the text without vowels in the associated `Label`.

private void btn_NoVowels_Click(object sender, EventArgs e)

- Handles the "No Vowels" button click event; removes vowels from input text and displays the result.

5. Add Evens



Button Text: "Add Evens"

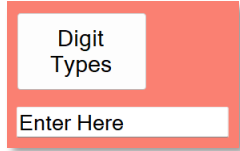
Instructions:

- Retrieve an integer input from the associated `TextBox`.
- Validate that the input is a positive integer greater than or equal to 2.
- Calculate the sum of all even numbers from 2 up to and including the input value.
- Display the sum in a message box.

private void btn_AddEvens_Click(object sender, EventArgs e)

- Handles the "Add Evens" button click event; calculates the sum of all even numbers up to a given value.

6. Digit Types



Button Text: "Digit Types"

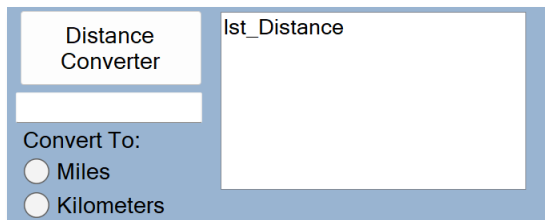
Instructions:

- Retrieve the input text from the associated `TextBox`.
- Validate that the input contains valid numeric characters (integers or decimals).
- Count and classify the digits into categories:
 - Total digits
 - Even digits
 - Odd digits
 - Zero digits
- Display the counts in a message box.

private void btn_Digits_Click(object sender, EventArgs e)

- Handles the "Digit Types" button click event; analyzes input for total, even, odd, and zero digits.

7. Distance Converter



Button Text: "Distance Converter"

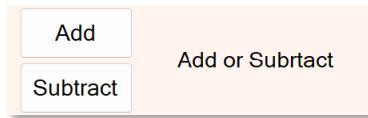
Instructions:

- Retrieve a numeric input (distance) from the associated `TextBox`.
- Validate that the input is a positive number.
- Use the selected `RadioButton` to determine the conversion:
 - If converting **miles to kilometers**, multiply by 1.60934.
 - If converting **kilometers to miles**, multiply by 0.621371.
- Display the converted value and unit in the associated `ListBox`.

private void btn_Distance_Click(object sender, EventArgs e)

- Handles the "Distance Converter" button click event; converts a distance between miles and kilometers based on user input.

8. Add



Button Text: "Add"

Instructions:

- Increment a counter variable each time the button is clicked.
- Update the associated Label (`lbl_AddSubtract`) to display the current counter value.
- Change the color of the Label:
 - Blue if the counter is zero or positive.
 - Red if the counter is negative.

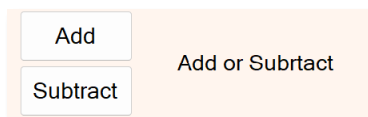
private void btn_Add_Click(object sender, EventArgs e)

- Handles the "Add" button click event; increments a counter and updates its display.

private void UpdateCounterDisplay()

- Updates the display of the counter value and changes its color based on whether the value is positive, negative, or zero.

9. Subtract



Button Text: "Subtract"

Instructions:

- Decrement the counter variable each time the button is clicked.
- Update the associated Label (`lbl_AddSubtract`) to display the current counter value.
- Change the color of the Label:
 - Blue if the counter is zero or positive.
 - Red if the counter is negative.

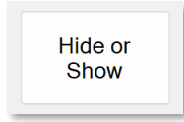
private void btn_Subtract_Click(object sender, EventArgs e)

- Handles the "Subtract" button click event; decrements a counter and updates its display.

private void UpdateCounterDisplay()

- Updates the display of the counter value and changes its color based on whether the value is positive, negative, or zero.

10. Hide or Show



Button Text: "Hide or Show"

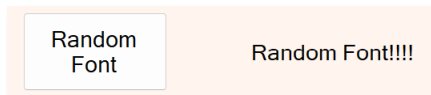
Instructions:

- Toggle the visibility of the associated `Panel` (`pnl_HideShow`):
 - If the panel is visible, hide it and update the button text to "Show Panel."
 - If the panel is hidden, show it and update the button text to "Hide Panel."

private void btn_HideShow_Click(object sender, EventArgs e)

- Handles the "Hide or Show" button click event; toggles the visibility of a specific panel and updates the button text.

11. Random Font



Button Text: "Random Font"

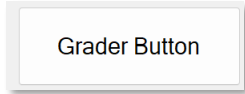
Instructions:

- Generate a random font size between 8 and 20.
- Randomly select a font style from available options such as:
 - Regular
 - Bold
 - Italic
 - Underline
 - Strikeout
- Generate a random color by creating random RGB values.
- Apply the random font size, style, and color to the associated `Label` (`lbl_RandomFont`).
- Ensure the changes are immediately visible to the user.

private void btn_RandomFont_Click(object sender, EventArgs e)

- Handles the "Random Font" button click event; assigns a random font size, style, and color to a label.

12. Grader Button



Button Text: "Grader Button"

Instructions:

- This button should only be worked on once you have ensured these items already work manually: `btn_VocabGen`, `btn_MonthlyPayment`, `btn_Backwards`, `btn_NoVowels`, `btn_AddEvens`, `btn_Digits`, `btn_Distance` (km and Miles))
- Simulate pressing all the other buttons in the form programmatically:
 - Populate the vocabulary list.
 - Calculate a sample monthly payment.
 - Reverse a sample text.
 - Remove vowels from a sample text.
 - Calculate the sum of even numbers for a sample input.
 - Analyze digits for a sample numeric input.
 - Perform distance conversion for sample values (km and miles)
- Display a summary message indicating that all tests have been completed.

private void btn_GraderButton_Click(object sender, EventArgs e)

- Handles the "Grader Button" click event; performs tests on multiple other buttons and functionalities in the form.

Rubric:

| Solution and Project (There is NO partial credit) (NOTE: UC represents uppercase and LC represents lowercase) | | |
|--|--|--------------------|
| The VS project file is present on the flash drive in a single folder with your contest ID | | 20 points |
| Button Execution (If the program does not execute, then the remaining items in this section receive a score of zero). The | | |
| btn_VocabGen Vocab Generator | | 20 points |
| btn_MonthlyPayment Monthly Payment | | 80 points |
| btn_Backwards Backwards | | 30 points |
| btn_NoVowels No Vowels | | 40 points |
| btn_AddEvens Add Evens | | 50 points |
| btn_Digits Digit Types | | 70 points |
| btn_Distance Distance Converter | | 100 points |
| btn_Add Add | | 30 points |
| btn_Subtract Subtract | | 30 points |
| btn_HideShow Hide or Show | | 40 points |
| btn_RandomFont Random Font | | 30 points |
| btn_GraderButton Grader Button | | 200 points |
| (NOTE: you must test the following: btn_VocabGen, btn_MonthlyPayment, btn_Backwards, btn_NoVowels, btn_AddEvens, btn_Digits, btn_Distance (km and Miles)) (NO PARTIAL CREDIT) | | |
| Invalid Input Handling Monthly Payment | | 10 points |
| Invalid Input Handling Add Evens | | 10 points |
| Invalid Input Handling Digit Types | | 10 points |
| Invalid Input Handling Distance Converter | | 10 points |
| Subtotal | | /780 Points |
| Source Code Review (Code must work to get credit.) | | |
| Program is properly commented (NOTE: in order to get the full credit of 50 points for the commenting requirement, you should place the comment flag in front of the comment in your code. The comment flag will precede the explanation. For example, if the flag is SC1, your comment should read as "//SC1..." in front of the part of the code being reviewed. Partial Credit may be awarded if some comments are present.) | | |
| A comment containing the contestant number is present at the top of the Form1.cs file | | 10 points |
| SC1: Vocab Generator: private void btn_VocabGen_Click(object sender, EventArgs e) Code that stores the list of Strings and populates a list box | | 10 points |
| SC2: Monthly Payment: private void btn_MonthlyPayment_Click(object sender, EventArgs e) Code that executes applying a mathematical formula for financial calculations and understanding order of operations. | | 30 points |

| | | |
|---|--|--------------------|
| SC3: Backwards: <i>private void btn_Backwards_Click(object sender, EventArgs e)</i> Code that uses a data structure to help reverse the string. | | 10 points |
| SC4: No Vowels: <i>private void btn_NoVowels_Click(object sender, EventArgs e)</i> Code that executes filtering vowels from a string | | 10 points |
| SC5: Add Evens: <i>private void btn_AddEvens_Click(object sender, EventArgs e)</i> Code that executes creating a loop to iterate over even numbers and accumulating a sum. | | 10 points |
| SC6: Digit Types: <i>private void btn_Digits_Click(object sender, EventArgs e)</i> Code that executes detecting and categorizing digits using conditional logic. | | 10 points |
| SC7: Distance Converter: <i>private void btn_Distance_Click(object sender, EventArgs e)</i> Code that executes branching logic based on radio button selection. | | 10 points |
| SC8: Add: <i>private void btn_Add_Click(object sender, EventArgs e)</i> Code that executes incrementing a variable and calling a method to update the UI using <i>private void UpdateCounterDisplay()</i> | | 10 points |
| SC9: Subtract: <i>private void btn_Subtract_Click(object sender, EventArgs e)</i> Code that executes decrementing a variable and calling a method to update the UI. <i>private void UpdateCounterDisplay()</i> | | 10 points |
| SC10: Hide or Show: <i>private void btn_HideShow_Click(object sender, EventArgs e)</i> Code that executes toggling the visibility of a UI component. | | 10 points |
| SC11: Random Font: <i>private void btn_RandomFont_Click(object sender, EventArgs e)</i> Code that executes dynamically creating and applying a new font with specific attributes. | | 20 points |
| SC12 (must be placed in these 7 locations; no partial credit): Grader Button: <i>private void btn_GraderButton_Click(object sender, EventArgs e)</i> Programmatically triggering button click events to automate testing. It must test the following: btn_VocabGen, btn_MonthlyPayment, btn_Backwards, btn_NoVowels, btn_AddEvens, btn_Digits, btn_Distance (km and Miles) | | 50 points |
| Subtotal | | /250 Points |
| Total Points | | /1030 |