

**Project 10 (40 points)**  
**Due: Friday, May 6<sup>th</sup> by midnight**

**Important:** Syllabus originally listed due date as WED, May 4<sup>th</sup>...given 2 extra days to submit *without penalty* – however, Project 10 **will only be accepted up to ONE day late** (10% penalty) – **nothing will be accepted after Sat, May 7<sup>th</sup>** (GTAs need time to grade the last project before the Final Exam)

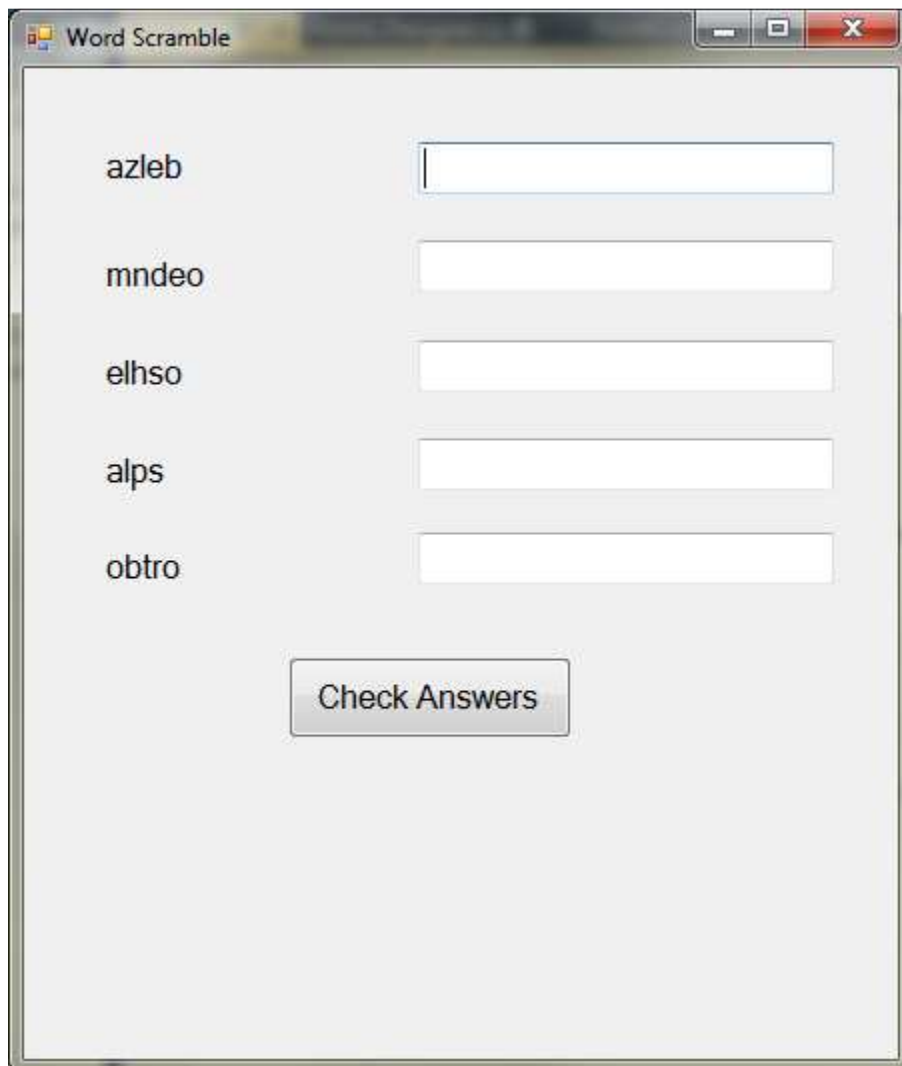
**Reminder:** ALL projects are **intended to be done individually. Don't share your code with anyone.** *If it is not your work, don't submit it as yours!* Refer to the policy within the course syllabus, which states, ***“If two (or more) students are involved in ANY violation of this policy, ALL students involved receive a zero for the assignment and the offense is officially reported to the KSU Honor Council. The second offense results in a failing grade for the course and possible suspension from the university (this decision is made by the K-State Honor Council).”***

---

**Assignment Description:**

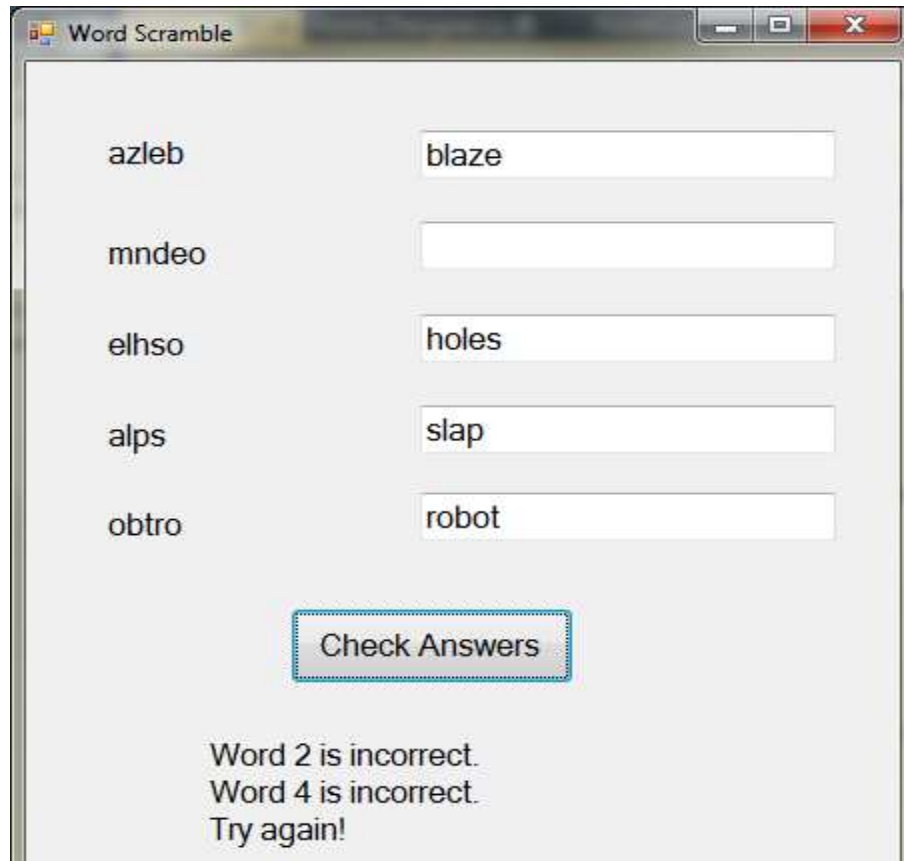
In Project 4, you created a console-based, non-object oriented *Word Jumble game*. For Project 10, you are to write a **graphical** word scramble puzzle game **using C#**. You will *randomly* pick five words with 4-5 letters each and display them for the user. The user will unscramble them, type their answers, and check the results. When your project starts, it should randomly display five different words.

Here is a possible puzzle:



The screenshot shows a window titled "Word Scramble" with a standard Windows title bar (minimize, maximize, close buttons). Inside the window, there are five scrambled words listed vertically on the left: "azleb", "mndeo", "elhso", "alps", and "obtro". To the right of each word is a corresponding empty text input field for the user to type the unscrambled word. At the bottom center of the window is a button labeled "Check Answers".

Suppose the user guessed several words and then gave up. Here is what would happen if they checked their results:



A screenshot of a Windows application window titled "Word Scramble". The window contains a list of five scrambled words on the left and their corresponding guess boxes on the right. The guesses are: "blaze" for "azleb", an empty box for "mndeo", "holes" for "elhso", "slap" for "alps", and "robot" for "obtro". Below the list is a "Check Answers" button. At the bottom, a message reads: "Word 2 is incorrect. Word 4 is incorrect. Try again!".

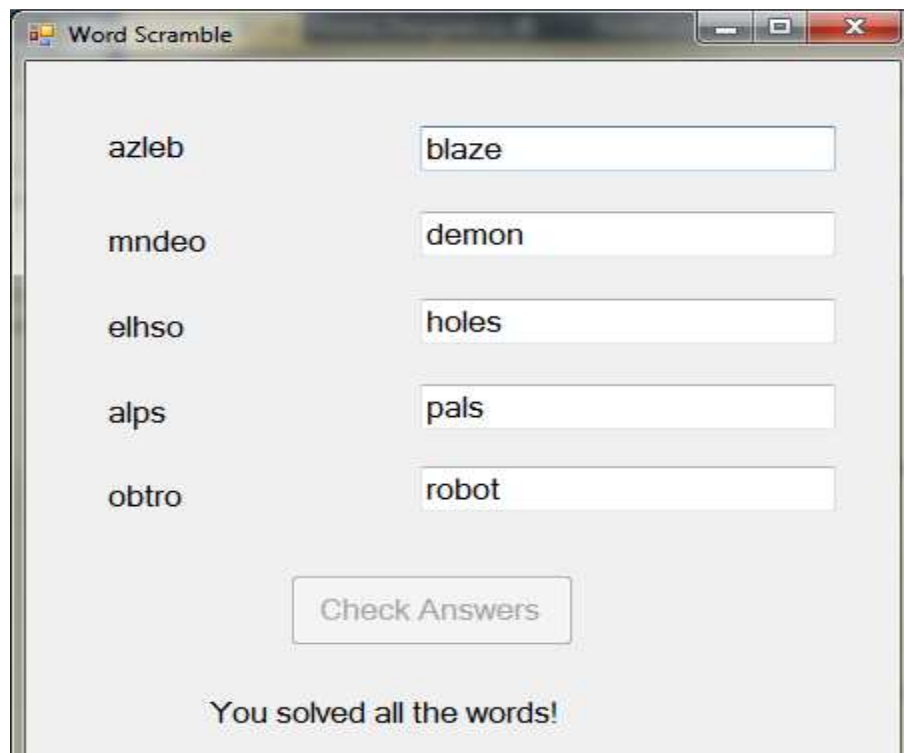
Scrambled Word	Guess
azleb	blaze
mndeo	
elhso	holes
alps	slap
obtro	robot

Check Answers

Word 2 is incorrect.  
Word 4 is incorrect.  
Try again!

(Note that Word 4 is incorrect even though "slap" is a valid jumble of "alps". If the user does not guess the exact word picked from the file, it is fine to count it wrong even if it is another possible solution.)

If the user successfully solves the rest of the puzzle and checks their answers again, they will see:



A screenshot of the "Word Scramble" application window after a successful solution. The guesses are now: "blaze" for "azleb", "demon" for "mndeo", "holes" for "elhso", "pals" for "alps", and "robot" for "obtro". The "Check Answers" button is now disabled. At the bottom, a message reads: "You solved all the words!".

Scrambled Word	Guess
azleb	blaze
mndeo	demon
elhso	holes
alps	pals
obtro	robot

Check Answers

You solved all the words!

## Implementation Requirements:

You may either modify your previously written Project 4 into C# or create the program from scratch. The only requirement is your program *must be written in C# using Visual Studio*. You should use the configuration shown in the screenshots above for setting up your display (*labels* for the random words, *text boxes* for the user guesses, a *button* for checking answers, and a *label* at the bottom for results.) For the most part you are free to implement your project in whatever way you choose, but here are a few additional requirements:

- You must have a method for randomly picking words
- You must have a method for randomly jumbling each of your selected words
- You must have “*click event*” method for your button
- If the user solves the puzzle correctly, disable the button (set `Enabled` property to `false`)

I suggest storing the correct words in an array and using an array of *labels* for the *jumbled words* and an array of *textboxes* for the *user guesses*. This will greatly simplify your code.

## Choosing and Jumbling Words

The five words for the puzzle should be randomly chosen from a file, `words.txt`, which contains a number of four- and five-letter words. Please download this file from Canvas and use it in picking your words. (The first line in the file is the number of words.)

**Note:** The `words.txt` file will go in the same folder as your `.exe` file, which is `...\bin\debug` within your Visual Studio project folder.

Additionally, each word should be jumbled to display its letters in a random order. (You should *NOT* have the same jumbling pattern for all words.) You may choose any approach for jumbling that you wish, but here is one possible algorithm (assuming “word” is the chosen string from the file):

- Create a *bool* array, one spot for each letter in the word
- This array will keep track of whether a letter has been added to the jumbled word yet (it will initialize all at *false* and set to *true* as each letter is used )
- Create a string, “*jumble*”, that starts as the empty string. Add letters onto *jumble* one at a time.

While not all letters have been added to *jumble*

    Randomly generate an index in word

    If that spot has not yet been chosen (check the *bool* array)

        Mark it chosen (*true*)

        Add the letter at that spot in word to the *jumble* string

**Important:** Note that a valid *jumble* of a word could be the original word itself. Prevent this from happening by using the following: *if jumbled word equal original word, re-jumble the word*

Random number generators in C# are very similar to Java:

```
Random r = new Random();  
int num = r.Next(5);
```

`num` would be randomly set to 0, 1, 2, 3, or 4.

**Helpful Hint:** Your main modifications/additions will be to the file called *Form1.cs*

**One possible** outline of *Form1.cs* might look like the following:

**using statements...**

```
namespace... {
    public partial class Form1 : Form {
        declare variables (like Label [ ], string [ ], TextBox [ ], Random, etc.)

        public Form1() {
            initializeComponent();

            variable assignment statements
            private method calls
        }

        private <return type> pickWords (parameters...) {
            ...
        }

        private <return type> jumbleWords (parameters...) {
            ...
        }

        private <return type> checkButton_Click (parameters...) {
            ...
        }
    } // end class
} // end namespace
```

---

**Documentation:** You must put a description of the project at the top of the file **and a description of the method at the top of each method.**

Please use this template for the top of the file:

```
/**
 * (description of the project)
 *
 * @author (your name)
 * @version (which number project this is)
 */
```

Please use this template for the top of each method:

```
/**
 * (description of the method)
 *
 * @param (describe first parameter)
 * @param (describe second parameter)
 * (list all parameters, one per line)
 * @return (describe what is being returned)
 */
```

---

### Submission:

To submit your project, **zip the entire project folder created by Visual Studio**. Go to “*Submit Projects Here*” folder on K-State Online. Select your lab time and upload the *zip* file. **Put your name and Project 10 in the description box.**

**Reminder:** It is the student’s responsibility to verify that the correct files are properly submitted. If you don’t properly submit the correct file, it will not be accepted after the 3-day late period.

The screenshot shows the K-State Online interface for Project 1. On the left is a navigation menu with links to Home, Announcements, Assignments (highlighted), Grades, People, and Files. The main content area shows 'Project 1' with a due date of Jan 30 by 11:59pm, 20 points, and a submission status of 'Submitting a file upload'. Below this, it says 'CIS 200: Project 1 (20 points)' and 'Due Friday, Jan 30th by 11:59pm'. A reminder note states: 'Reminder: Programs submitted after the due date/time will be penalized 10% for each day the project is late (not...'. On the right, a 'Submission' box shows a green checkmark and 'Turned In!' status, with the time 'Jan 20 at 2:54pm'. It includes links for 'Submission Details' and 'Download Project1.zip', and a comment from 'John Doe - Project 1'.

**Grading: Programs that do not compile will receive a grade of 0. Remember to submit the ENTIRE Visual Studio Project as a .zip file**

Requirement	Points
Correctly create GUI shown on p.1 of the instruction sheet with correct Frame Title, 5 labels, 5 text fields, and correctly labeled button. (Note: GUI does <i>NOT</i> have to work to get credit for this portion.)	10
Includes working <i>method</i> for randomly picking words from given text file.	5
Includes working <i>method</i> for randomly jumbling each word and displaying jumbled word in the GUI. Re-jumbles the word if <i>jumbled word</i> = <i>original word</i>	5
Includes working “ <i>click event</i> ” method for your button	3
Game works as described in instructions – displays which words are incorrect or “ <i>solved all words</i> ” message and disables the button	13
<b>Project Submission</b> (entire Visual Studio Project with name and Project # in description box)	2
<b>Documentation</b> – includes documentation on <i>EACH file</i> and above <i>EACH method</i>	2
<b>Total</b>	<b>40</b>