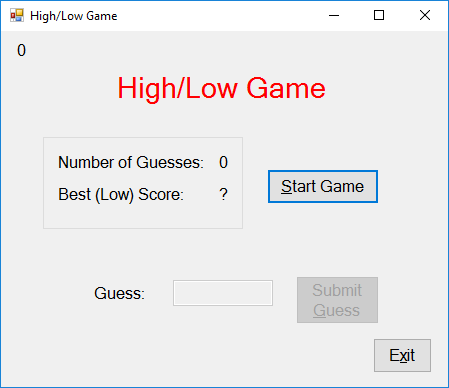
**Final Project  
CIS 146**

Follow these instructions carefully. I am the client/designer – you are the coder. **Do not vary from the design without client/designer permission**. To solve the problem, I will provide you with an IPO, TOE, screen design and pseudocode. Each line of pseudocode represents a single or maybe two commands. Do not make this assignment difficult. **I am here to help.** Historically, a simple challenge that causes you frustration might take me no more than five minutes to explain.

**Overview**

In this final we will be putting the pieces of the puzzle we have learned together. You will be using variables, math, conditional statements and the Windows forms design techniques we have covered to create a High/Low game.



When the user starts the game, your program will generate a random number between 0 and 1000. I will provide the code to generate a random number. For design and testing purposes, I am displaying the random number the computer selects in the top left corner of the form. I would normally hide or remove the label before deployment. In our case, leave the label visible so it is easier for me to grade and you to test.



As the user enters a guess, a message box indicating “Too High”, “Too Low”, or “You guessed it in *x* attempts” displays in a message box and “x” is replaced with an integer. The Number of Guesses label indicates the number attempts the user has made.

If the player’s score is the lowest (best), that score is placed in the Best Score label.

**IPO**

|  |  |  |
| --- | --- | --- |
| Random Number (computer) | Computer generates a random number between 0-1000.  User attempts to guess the number. Display number of attempts. If guess > random then “Too High” else if the guess < random then “Too Low” else the user guessed the number. If their number of guesses is less than Best Score, replace it and display the new best score and reset the game. | “Too High” or |
| User’s Guess | “Too Low” or |
|  | “You Guessed it” |
|  |  |
|  | Lowest Score |
|  | Number of Guesses |
|  |  |
|  |  |
|  |  |

**TOE**

|  |  |  |
| --- | --- | --- |
| T | O | E |
| Form | frmHighLow |  |
| Exit Button | btnExit | *Click* |
| Start Game Button | btnStart | *Click* |
| Best Score Label | lblBestScore |  |
| Number of Guesses Label | lblNumGuesses |  |
| Guess Text Box | txtGuess |  |
| Random Number Label (temporary) | lblRandom Number |  |
| Submit Guess Button | btnGuess | *Click* |

**TOE Pseudocode**

**Start Game Button Click**

Generate a Random number to the Random Number Variable

Set Guesses Variable to 0

Disable the Start Button

Enable the Guess Button

Enable the Guess Text Box

Set Focus to the Guess Text Box

**Guess Button Click**

Increment Number of Guesses variable by 1

Display number of Guesses in Number of Guesses Label

if the guess in the text box is not numeric (I will give you this code)

{

Display a Message Box "Numbers Only"

Clear the guess text box

Set focus to the guess text box

}

else

{

Move the value in the guess textbox into a guess variable.

if the guess variable is less than the random number

{

Display in a message box "Too Low";

Clear the guess text box

Set focus to the guess text box

}

else if the guess variable is greater than the random number

{

Display in a message box "Too High";

Clear the guess text box

Set focus to the guess text box

}

else (it was guessed)

{

Display in a message box "You Guessed it in x attempts.");

Disable the guess text box

Disable the guess button

Enable the Start Button

if the current user's number of guesses < lowest score

{

Set the best score integer to the user's number of guesses

Display the new best score in the best score label

Reset the current number of guesses label to 0 (ready for new game)

Reset the current number of guesses variable to 0 (ready for new game)

Clear the number in the guess text box

}

}

}

**Exit Button Click**

Quit Program

**Getting Started**

1. Build the screen as close to the screen design as I provided. Use the object names I provide in the TOE. Take note that the Number of Guesses and Best Score sit on a Group Box.
2. The Submit Guess button and Guess text box should be disabled at design time. You can’t submit a guess until the game has started.
3. Set your form’s tab order
4. Make the Submit button your form’s Accept button
5. Make the “S” in “Start Game” the hot key.
6. Make “G” in “Submit Guess” the hot key.
7. Make the “x” in “Exit” the hot key.
8. Although not included in the IPO or TOE, it will make your coding easier if you maintain all your numeric values in variables and simply display those variables back to the screen. Create three class level integer variables – one to hold the generated random number, one for number of guesses and finally best score. Set the value of the best score initially to 99 as a primer. We will display a “?” when the game starts. Once someone gets a score below 11, the ? will be replaced with new low score.
9. Code your Start Button Click Event handler. Use this code at the start to generate a random number. Keep in mind my example assumes you used *intRandomNumber* as your variable.

Random rnRandomNumber = new Random();

intRandomNumber = rnRandomNumber.Next(0, 1000);

*Note: You might find it easier to replace all the pseudocode references to variables and objects with the actual names you use.*

1. Add the following code (function or value returning method) above the second to the last bracket in your code.

private bool isNumberInt(string strInput)

{

try

{

int intValue = int.Parse(strInput);

return true;

}

catch (Exception)

{

return false;

}

}

This function determines if a value in a text box is numeric. It can be used in your Guess Click event handler like this:

if (!isNumberInt(txtGuess.Text))

{

MessageBox.Show("Integers Only. Retry.");

txtGuess.Focus();

}

1. Upon completion, test your program thoroughly. Reread the instructions to make sure you didn’t skip something. Zip your work and turn it into the drop box.

**Rubric**

|  |  |  |
| --- | --- | --- |
| **Screen Design** | |  |
|  | 1. Match L-n-F | 15 |
|  | 1. Named Objects | 4 |
|  | 1. Tab Order | 3 |
|  | 1. Hot Keys | 3 |
|  | 1. Accept Button | 3 |
|  |  |  |
| **Code** |  |  |
|  | 1. Does it Function w/o Error (can I force an error?) | 12 |
|  | 1. Class Level Variables | 4 |
|  |  |  |
|  | ***Coding***   1. Start | 14 |
|  | 1. Guess Button | 14 |
|  | 1. Guess Text Box Leave | 14 |
|  | 1. Exit | 2 |
|  | 1. Neatness (Spaced, Extraneous code?) | 6 |
|  | 1. Followed Other Instructions (Code Design) | 8 |

**Extra Credit (Maximum score of 110 points where 95 is a 4.0)**

**If you do extra credit you must indicate so at submission. You must have completed the above instructions with 90% or better (unless otherwise stated) before attempting extra credit. Turn in a modified TOE with your extra credit if it impacts functionality.**

* Turn assignment in 3 days early for 3 points of extra credit. Work must be 90% or better to receive the early bonus.
* Add a counter allowing the user to have a maximum of 10 attempts before losing. +3
* Add a game appropriate image to your app. +1
* Add sound to your app. Replace message box with audio. +3
* Capture and display the best score **and** the best scorer’s name. +2
* Save the best score so that each time you start the program the best score is retained. Hint: Google: C# **Settings**. +3