NAME _____

EET 2259 Lab 4
The String Data Type

OBJECTIVES

-Understand the differences between numeric data, Boolean data, and string data.

-Write programs using LabVIEW's string controls and indicators, string constants, string functions, combo boxes, and rings.

Part 1. Review of Analog and Digital I/O

The main topic of this lab is the string data type. But first let's get some more practice using the myDAQ to do some analog and digital I/O. To write this first VI, **you'll need to recall what you learned in Labs 1, 2, and 3 about the DAQ Assistant, the myDAQ, and the red trainer.**

1. Create a VI whose front panel contains a numeric indicator labeled **Voltage** and an LED labeled **Voltage > 3 V**. On the block diagram, place a DAQ Assistant that reads and displays the value of the voltage being produced by the trainer's power supply. The front panel's LED should light up if the voltage is greater than 3 V, but the LED should stay dark otherwise.
2. Modify your VI's front panel by adding a push button labeled **Check Voltage?**. Wire the block diagram so that when this push button is turned on, the LED behaves exactly as it did above in Step 1. But when the push button is turned off, the LED should stay dark, no matter what the value of the input voltage is.
3. Now replace the VI's push button with one of the trainer's data switches. In other words, delete the push button from the front panel and wire up one of the trainer's data switches so that it behaves the way the push button behaved in Step 2. Save this VI as **Lab4AnalogDigitalInput.vi**, and **show me your working program**.

_____

Part 2. String Data

A **string** is a sequence of characters. The characters could be letters, numerals, punctuation marks, and so on. When we write about a string, we enclose the string in quotation marks. For instance, "Abe Lincoln" is a string containing eleven characters. Ten of those characters are letters, and the other one is a space. The quotation marks are not part of the string. They just show us where the string begins and where it ends. As another example, "Dayton, OH 45402" is a string containing sixteen characters. Eight of these characters are letters, five are numerals, two are spaces, and one is a comma.

How many characters does the string "http://www.sinclair.edu" contain? Write your answer below.

_____

In the two previous labs you've studied numeric data and Boolean data. Now we'll look at string data. Just as LabVIEW has controls, indicators, constants, and functions for numeric data or Boolean data, it also has controls, indicators, constants and functions for string data.

Part 3. String Controls and Indicators

String controls in LabVIEW let the user enter strings. String indicators display strings to the user. Under the **Modern** category of the Controls palette, click the **String & Path** icon to access a palette containing a string control, a string indicator, and a few other controls and indicators. For practice with these, perform the following steps.

1. Create a VI whose front panel has two string controls labeled **First name** and **Last name**, a string indicator labeled **Selected name**, and a horizontal toggle switch labeled **First name or last name?** Wire the block diagram so that when the user runs the program with names typed into the string controls, the string indicator displays either the first name or the last name, depending on the position of the switch. (Hint: You'll need to use one of the functions from the Comparison palette that you learned about in the previous lab.) Save this VI as **Lab4SelectName.vi**, and **show me your working program**.

Part 4. String Constants

Just as a LabVIEW block diagram can contain numeric constants and Boolean constants, it can also contain string constants. Here's how to place a string constant on a block diagram: under the **Programming** category of the Functions palette, click the **String** icon to access a palette containing about fifteen string functions, seven string constants, and icons for some sub-palettes with more string functions. Notice that string constants have a pink border. For practice with string constants, perform the following steps.

1. Create a VI whose front panel has a horizontal toggle switch labeled **Coming or going?** and a string indicator labeled **What to say**. Wire the block diagram so that the string indicator displays the word "Hello" if the switch is switched off, and displays the word "Goodbye" if the switch is switched on. Save this VI as **Lab4SelectGreeting.vi**, and **show me your working program**.

You'll like the next program if your name is Mike.

1. Create a new VI whose front panel has two string controls labeled **First name** and **Last name**, and an LED labeled **It's Mike!** Wire the block diagram so that the LED lights up if the user enters "Mike" into the string control labeled **First Name**, but the LED stays dark the rest of the time. (Hint: For this step and the following steps you'll need to use some of the Comparison and Boolean functions that you learned about in the previous lab.)
2. Change the LED's label to **It's Mike Smith!** and then modify the VI so that the LED lights up if the user enters "Mike" for the first name and "Smith" for the last name, but the LED stays dark the rest of the time.
3. Change the LED's label to **It's Mike Smith or Mike Jones!** and then modify the VI so that the LED lights up if the user enters "Mike" for the first name and "Smith" or "Jones" for the last name, but the LED stays dark the rest of the time. Save this VI as **Lab4NameCheck.vi**, and **show me your working program**.

Part 5. Dialog Boxes

To really get the user's attention, you can display a message in a dialog box instead of in a string indicator. You'll find dialog boxes on the **Functions > Programming > Dialog & User Interface** palette.

1. **NOTE:** After you write this program, run it using the Run button, **not the Run Continuously button**. If you try to run it continuously, you'll get into an infinite loop and you'll have to use the Ctrl-Alt-Delete keys to get out of it.
2. Open your program called Lab4SelectGreeting.vi and save a new copy of it under the name Lab4GreetingDialog.vi. Then remove the string indicator from the block diagram and replace it with a **One Button Dialog**. When you run the program, a dialog box should display the appropriate greeting.
3. Read LabVIEW's Help information about the One Button Dialog to figure out how to change the label on the dialog box's button from **OK** to **Okey-Doke**. Save this VI as **Lab4GreetingDialog.vi**, and **show me your working program**.

Part 6. String Functions

Recall that LabVIEW's numeric functions let you perform operations on numbers, such as adding, taking square roots, and so on. LabVIEW's string functions let you perform operations on strings, such as:

- finding the number of characters in a string
- converting the letters in a string to uppercase or lowercase
- combining two or more strings into a single string (The name for this is **concatenating.**)
- and lots more.

These functions are found on the same palette where you found string constants. Some of these functions have many inputs. You may need to study LabVIEW's Help information to figure out how to use them. The following program uses some of LabVIEW's simpler string functions.

1. Create a VI whose front panel has two string controls labeled **First name** and **Last name** and a string indicator labeled **Full name**. Wire the block diagram so that when the user runs the program with names typed into the string controls, the string indicator displays the full name, with a space separating the first name from the last name.
2. Modify this VI by adding an LED labeled **That's a long name**. Wire the block diagram so that the LED lights up if the full name (including the space) contains 20 or more characters, but the LED remains dark otherwise.
3. Modify this VI by adding a horizontal toggle switch labeled **First name first or last name first?** Wire the block diagram so that if the switch is set to **First name first**, the full name will be displayed as the first name followed by a space followed by the last name. But if the switch is set to **last name first**, the full name will be displayed as the last name followed by a comma followed by a space followed by the first name. The LED should continue to work as it did above, and you *should not count the comma* as a character in the name.
4. Modify this VI by adding a push button labeled **Display in uppercase?** If this push button is off, the VI should work exactly as it did above. But if the push button is on, the full name should be displayed in all uppercase letters. (The toggle switch and the LED should continue to work as they did above.) Save this VI as **Lab4StringFunctions.vi**, and **show me your working program**.

Part 7. Combo Boxes

Another control on the **String & Path** palette is called a **combo box.** It's handy when the user needs to be able to enter a string from a limited number of choices. A combo box is similar to a string control, but the user, instead of having to type the text, can choose from a list of pre-defined strings by clicking the combo box's "drop-down arrow." For this next example, suppose the user needs to choose one of the ten colors in the resistor color code.

1. Create a VI whose front panel has a combo box labeled **Resistor color band** and a string indicator labeled **Output String**. Right-click the combo box to open its shortcut menu, and choose **Edit Items…**. Insert the ten color names, in the following order, and capitalize the first letter in each word: Black, Brown, Red, Orange, Yellow, Green, Blue, Violet, Gray, White.

2. On the block diagram, wire the combo box to the string indicator. When you run the program, select a color by using the drop-down arrow, and notice that the selected color appears in the string indicator.

3. By default, a combo box will let the user type in strings that do not appear on the list. For example, notice that in your program you can type "Magenta" into the combo box. But if you don't want the user to be able to type in his own strings, LabVIEW lets you prevent him from doing this. To see how, right-click the combo box and remove the check mark from **Allow Undefined Strings**. Now notice that you can no longer type in the word "Magenta."

4. On the front panel, add a horizontal toggle switch labeled **Uppercase or Lowercase?**. Add code on the block diagram so that the selected string is displayed either in uppercase or in lowercase, depending on the switch's position.

5. Save this VI as **Lab4ComboBox.vi**, and **show me your working program**.

Part 8. LabVIEW's Three Major Data Types Compared

The following table summarizes some facts about the three major data types in LabVIEW. The table has one column for the numeric data type, one column for the Boolean data type, and one column for the string data type. You should memorize the information in this table.

	Numeric	**Boolean**	**String**
Controls	Numeric control, slides, knob, dial.	Switches and buttons.	String control, combo box.
Indicators	Numeric indicator, gauge, meter, progress bars.	LEDs.	String indicator.
Values	Many possible values, such as 0, 1, 2, … May be integer values or floating point values.	Only two possible values: true and false.	Many possible values, such as "Howdy partner" or "July 4, 1776"
Functions	Add, Subtract, Multiply, Divide, Square Root, Absolute Value, Round, and many other numeric functions.	And, Or, Not, and other logical functions.	To Upper Case, Concatenate, String Length, and many other string functions.
Color on Block Diagram	Blue (for integers) or orange (for floating-point numbers).	Green.	Pink.

Part 9. Rings

Are you ready for another type of control that looks a lot like a combo box but behaves very differently? Remember from above that a combo box's data type is string, and therefore combo boxes are colored pink on the block diagram. On the other hand, **text rings** and **menu rings** look similar to combo boxes on the front panel, but their data type is numeric instead of string. So although they let the user choose from a list of strings, the value they produce on the block diagram is a number, not a string. This can be very useful, as you'll see below.

1. Create a VI whose front panel has a text ring labeled **Resistor color band** and a numeric indicator labeled **Value**. (Text rings are found on the **Ring & Enum** palette. Menu rings are almost exactly the same; they just look a little different on the front panel.)

2. Right-click the text ring to open its shortcut menu, and choose **Edit Items…**. Insert the same ten color names as above: Black, Brown, Red, and so on.

3. As you insert these values, notice that each color is automatically assigned a numeric value. Use the **Move Up** and **Move Down** buttons to rearrange the colors until the values are correct for the resistor color code. (For example, Black = 0.)

4. On the block diagram, wire the text ring to the numeric indicator. When you run the program, select a color, and the selected color's value will appear in the numeric indicator.

5. Save this VI as **Lab4TextRing.vi**, and **show me your working program**.

Next, let's have a little fun with picture rings. A picture ring is like a text ring, except that instead of letting the user choose from a list of strings, it lets the user select from a list of pictures.

1. Create a VI whose front panel has a picture ring labeled **Resistor color band** and a numeric indicator labeled **Value**.

2. To add a picture to a picture ring, you must first have a picture on the Windows clipboard. So start your Web browser and do a Google image search for something black—maybe a black cat or a black Porsche. When you find an image that you like, copy it to the Windows clipboard by right-clicking it and selecting either **Copy** or **Copy Image**, depending on which Web browser you're using.

3. Back in LabVIEW, right-click your picture ring on the front panel and select **Import Picture from Clipboard**. You should now see the black image inside the picture ring. You may have to resize the ring to see the entire image.

4. Next, go back to your Web browser and find a good image of something brown. Copy it to the Windows clipboard.

5. Back in LabVIEW, right-click your picture ring and select **Import Picture After**. You should now have two pictures in the picture ring, and you should be able to choose between them simply by clicking and choosing the one you want.

6. Repeat the previous steps for the remaining colors in the resistor color code, in correct order.

7. On the block diagram, wire the picture ring to the numeric indicator. When you run the program, select a color, and the selected color's value will appear in the numeric indicator.

8. Save this VI as **Lab4PictureRing.vi**, and **show me your working program**.

Next you'll write a program that converts three-band color codes to resistor values.

1. Create a new VI whose front panel has a numeric indicator labeled **Resistor Value**. Also copy and paste the text ring from your program named Lab4TextRing.vi. (You could also recreate this text ring from scratch, but it's faster to copy and paste it.) Change its label to **Color Band 1**.

2. Paste two more copies of the text ring, and label these **Color Band 2** and **Color Band 3.** Wire the block diagram so that when the user runs the program with three colors entered into the text rings, the numeric indicator displays the correct resistance, in ohms. For example, if the user enters blue-gray-red for the three colors, the numeric indicator should display 6800.
    - Hint #1: I can think of at least two ways to do this using what you know about LabVIEW.
        - One way is to leave the text rings as they are and use a few math functions on the block diagram to calculate the total value based on the three color values. To do it this way, you'll need the **10^x** function on the **Functions > Mathematics > Elementary > Exponential** palette.
        - The other way, which requires fewer math functions, is to make the text rings produce different values. To do this, right-click a text ring and select **Edit Items…**. Then uncheck the **Sequential values** checkbox, and type in your own numerical value for each color. You may find that you need to change a text ring's representation from U16 to a numeric data type that can hold bigger values.
    - Hint #2: If you find this difficult, start by concentrating on the first two color bands, instead of trying to do all three at once.

3. In Lab 2 you learned how to configure a numeric indicator to display its number using engineering prefixes, such as k for kilo-. Do this with your **Resistor Value** indicator. Now for example, if the user enters blue-gray-red, the numeric indicator should display 6.8k.

4. Save this VI as **Lab4ColorsToNumbers.vi**, and **show me your working program**.

******* This lab had 10 named programs for me to check. If you didn't finish all of these during class, finish them after class. Then upload all 10 programs to the website by the due date. Also turn in your lab sheets at the beginning of class.********