DECISIONS, DECISIONS, DECISIONS

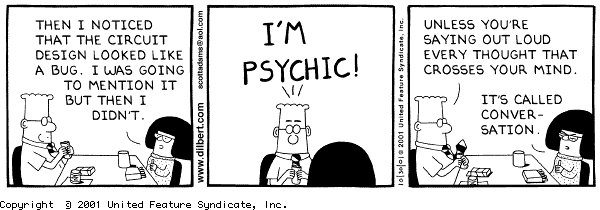
# **LAB 4**

**What You’ll Learn In This Lab:**

* To design and test simple circuits
* To create conditional statements in Python
* To experiment briefly with color distances and data input in Python
* To understand and experience the relationships between circuits and Boolean conditions
* To create conditional statements using the Finch

**Getting Ready:**

* Read Chapter 4 of CSI
* Bring the following to lab:
  + Lab 3 Worksheet, ready to turn in at beginning of lab!
  + CSIS-110 Lab Manual
  + Both textbooks
  + Something to write with!
  + The Lab 4 Worksheet to write answers on



**Details**

*This section of each lab will involve practicing and experimenting with various aspects of*

*our computer systems – editing, printing, mail, techniques for making your life easier, saving your projects.*

None this week.

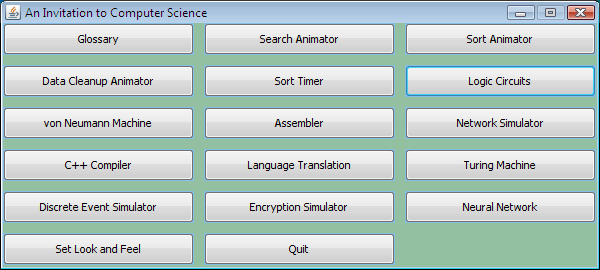
**Enhancing Your Computer Science Knowledge**

*This section of each lab will involve problems and software that will augment your understanding of concepts that are fundamental to Computer Science.*

For this and some upcoming labs, you will be using the *Invitation to Computer Science Laboratory Manual* and accompanying software. The Department has made these manuals available to you so that you don’t have to purchase them; **we only ask that you do not write in them, leave them in the lab, and take care of them so that we can make them available to future students as well!**

**Designing and Testing Simple Circuits**

**From the start menu, open the “Computer” window.** Navigate to the 110 student common folder; within it, there is a folder icon labeled **Invitation**. Double click on it, and then double click on the **Invitation.jar** file in order to open the software. After you click on “continue”, you’ll see a menu of different components.

Select “Set Look and Feel”, and click the Windows radio button on the right, then OK. Start the “Logic Circuits” component.

Get a lab manual (orange and white book) from your instructor, and read Lab Experience 7 (p. 69-73) in the book. Use the manual to do exercises in the text as described below.

Notes:

* The software package only lets you place switches in the left-most grid cells.
* In the diagrams, assume that the top switch is A, and the bottom is B.
* When you draw connections between switches, **connect from left to right**.

Use the Example circuit as described on p. 70. To open this file, go to the CSIS 110 Common folder, and then go to Invitation|Examples|Example.cir. **Do exercise 7.1, and draw the truth table on your worksheet.**

In plain English (20 words or less), **describe what the circuit of Exercise 7.1 does.** Don’t refer to any components of the circuit or truth table – we want a general description!



**Do exercise 7.2, and draw the corresponding truth table.** (Note: don’t save this circuit)

**In plain English (20 words or less), describe what the circuit in 7.2 does.** Don’t refer to any components of the circuit or truth table – we want a general description!

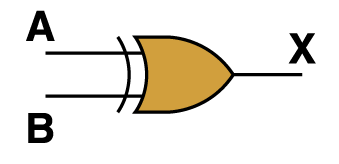
The following boxed section is 5 points Extra Credit; skip it for now then come back to it at the end of the lab if you have time!

**Do exercise 7.4.** For this exercise, you will create a new circuit that will perform exactly the same way as the original Example.cir, but will consist of only *one* AND gate and only *one* NOT gate. **Draw the truth table, and write the corresponding Boolean expression.**



**Print** the circuit of exercise 7.4 and attach the printout to your worksheet.

Create a circuit that represents the Exclusive Or operation – the symbol for it looks like this:



This is not a basic gate, because it can be created from the basic three (AND, OR, NOT). This gate outputs a true if **exactly one** of the inputs is true, and false otherwise.



**Draw** the truth table and boolean expression.

**Print** your Exclusive Or gate.

**Extending and Expanding**

*This section of each lab will involve using Python to answer questions and solve problems.*

**Modifying Pixels in a Range**

**Copy** the file “lab4Expanding.py” from the lab 4 common folder into your “csis110” folder. **Start** JES and **load** your copy of the file into JES. This is a program that manipulates a portion of the picture “bigben.jpg” from the MediaSources folder. In the command area, use **setMediaPath()** to set the media path to look for files in the MediaSources folder. Then run the program to see what it does by typing the command **bigbenColors()**.

You’ll see Big Ben with a large red rectangle in the middle. To make the rest of this exercise easier, type the following in the command area:

**explore (makePicture (getMediaPath("bigben.jpg")))**

You should keep that explorer window open until you’re finished with the lab! The explorer window is useful for determing both coordinates and colors.

Your first task is to alter the program so that just the square containing the clock face is red. It doesn’t have to be perfect, but get it as close as you can. Don’t forget to follow the general programming algorithm: Fix, Save, Load, and Test, then repeat until it works!

Your next task is to change the program in two ways: (1) Within the square, only the white parts of the clock’s face should change color; and (2) (this is the easy part) they should change to green, not to red (in honor of Siena College’s colors!). To do this, you will need a couple new Python concepts: the “if” statement and the “distance” function for colors. Check out Lines 10 and 11 of Program 49 on page 117 of your Python book. Line 10 gets the whole color (red, green, and blue values) of a pixel at once. That color is then compared to the color “brown” to see how close it is to brown. If it’s “close enough” (within 50), then the program changes the color of that pixel.

You want to do something very similar with “bigbenColors”: use “if” and “distance” and “getColor” to change just the white pixels within your range to green. You may have to experiment some with the proper distance to use (try starting with 50.0, then adjust it from there).



**Once you’ve turned the clock face green, get your instructor’s signature on your worksheet.**

One more task with Big Ben before we move on! For this one, you want to turn both the white and gold pixels to green (again, within the range of the square only). Note that “gold” is not a built-in color name in Python, so you will need to create a color to use, using the explorer window and the “makeColor” command of Python. You should add a line something like the following to your program (make sure you use actual numbers in the parentheses of course, and be careful where in the program you put it – do not make your program do any needless work):

**gold = makeColor ( …)**

Now you can use “gold” just like you used “green” and “red” earlier. **NOTE**: Your solution should use just one “if” statement!



**Once you’ve also turned the gold portions green, get your instructor’s signature on your worksheet!**



**Print “lab4Expanding.py” and attach it to the back of your lab report.**

**Conditional Statements using the Finch**

Now it’s time for what you have all been waiting for, the Finch! Your Finch gets slightly cranky when placed on his tail and very cranky when he is shaken. Lucky for us the Finch has the functions isBeakUp() and isShaken() to help us determine what state our Finch is currently in. You will complete a function called happyState() that will determine the finch’s current state and allow him to communicate his happiness or unhappiness. Your Finch will communicate using the saySomething() and setLED() functions. Finally, you will use a while loop so that the Finch keeps communicating his happiness or unhappiness until he is turned upside down, at which point the program will end. The isFinchUpsideDown() function will come in very handy! Some of the code has been written for you; your job is to finish it. Open the file “lab4FinchFun.py” from the common folder in JES-Finch and save it to your own folder. There are comments in the code to guide you about where you need to add statements and what exactly you need to add.

**Once you have completed your code and tested it, demo your Finch for your instructor and get their signature on your worksheet!**

** Print “lab4FinchFun.py” and attach it to the back of your lab report.**

**Gates and Circuits in Python**

In this section you will complete a program that simulates an “XOR” gate; this is the Exclusive Or gate that you created in the circuit program earlier in lab.

**Copy** the file “lab4XOR.py” from the common area into your “csis110” folder and **open** your copy of the file in JES. This file contains part of a program that will get two binary truth values (1 or 0) from the program’s user and will display the results of XOR applied to those input truth values. Read the comments in the program. You are to complete the program so that it displays a picture with a white background and either “TRUE” in green text or “FALSE” in red text, corresponding to the result of XOR applied to the two input values.

Test your program exhaustively.



Describe what exactly you did to test it “exhaustively” (hint: that doesn’t mean test it until you are tired.)

**Show** your instructor your XOR program for signature.



**Print** “lab4XOR.py” and attach it to the Worksheet.

**Reflection:**

*The discussion questions in this section of each lab are meant to make you think critically and creatively about some of the things you did earlier in the lab. Your answers to these questions must not be written on the lab handout, but on separate sheets of paper attached to the end of your lab report. Your answers must not be handwritten, and you will be graded on all aspects of your answer (correctness, use of proper terminology, readability, use of complete sentences only, etc.). In general you are expected to write at least one or two paragraphs in answer to each question.*

We have been working with a system called Boolean logic. This is sometimes called 2-valued logic, because there are two possible values for everything: True and False. As we know, one reason for using this system is that the hardware devices used to build computers have 2 easily-distinguished states (on and off). Suppose now that each transistor had 3 easily-distinguished states and that therefore we wanted to build gates out of these transistors.

Describe the logical system that would result from this. What would you call this system? What would its three possible values be called, and what would they mean (we all know what True and False mean, for example)? How would the three basic operations (and, or, not) change? Would this system have any advantages over the two-valued system to which we are accustomed? Would you need new operations? If so, what would they look like? Be specific and use examples. Also use your imagination – there is no single right answer.