References:
1. RobotBASIC: http://robotbasic.org

Introduction:
This lab continues from the previous lab in exploring use of RobotBASIC language constructs and features to solve problems.

Objectives:
- To gain understanding and use of basic programming constructs as found in the BASIC language.
- To gain experience in applying flowchart design to a problem solution.
- To further explore RobotBASIC built-in features.

Part 1: Resistor Color Code Helper

Problem statement: design and implement a computer program in RobotBASIC that will achieve the following requirements:

i. display a greeting message to the user
ii. prompt the user to enter a numeric value that represents a resistor value
iii. determine the resistor color code for that value
iv. display the resistor color code in English words, and
v. show a graphic representation of the resistor

You may assume a 4-band resistor and ignore the tolerance band for this exercise.

Procedure:
1. Requirement i can easily be accomplished with RobotBASIC’s Print command as seen in the previous lab.
2. Requirement ii can be accomplished with the Input command. You’ll need to specify a variable name for storing the inputted value. The following steps will refer to this variable as rv (bold here indicates variable names). Use the version of the Input command that includes a prompt string.
3. Requirement iii will need a solution strategy to extract the first 2 digits (bands) and the number of following digits to determine the multiplier (3rd band). Let’s try this recipe:
   a. initialize digits to 0, digit1 to 0 and digit2 to 0
   b. copy current digit1 value to digit2
   c. extract the least significant digit of rv into variable digit1; i.e. digit1 = rv MOD 10 (the remainder after dividing rv by 10)
   d. rv = rv / 10 (discard rv’s least significant digit)
   e. increment the digit counter digits
   f. repeat b...e until rv becomes 0
   g. multiplier = digits – 2 (so 1st two numeric digits are not included in the 3rd band)
4. For our initial attempt at requirement iv, let’s simply display (Print) the values of digit1, digit2, and multiplier so we can verify proper results for various test runs.
5. Sketch a flowchart of your proposed solution to the above steps. Use proper flowchart symbology.
6. Start a new RobotBASIC program and using your flowchart as a guide, implement steps 1 through 4 above. Step 3f will require use of BASIC’s repeat...until construct. Refer to the Helpfile section “Flow Control Statements” for the description and example. Verify proper operation with various test input values. Save
your program to an appropriate filename. If you need instructor help on this, make sure you first have your flowchart prior to calling for help.

7. Now we revisit requirement iv’s specification to output the color bands in English words. Since we have single-digit numeric values in `digit1`, `digit2`, and `multiplier`, we can use these as index values to access appropriate word strings in an array of `ColorNames`. Add the following Data statement to the beginning of your program. This statement creates a 10-element array of color names initialized to appropriate string values according to the resistor color code.

   ```plaintext
   Data ColorNames;"Black","Brown","Red","Orange","Yellow","Green","Blue","Violet","Gray","White"
   ```

8. Next, to output the resistor’s color code in English words, add the following lines to the end of your program.

   ```plaintext
   Print "Color code is ",
   Print ColorNames[digit1],"-",
   Print ColorNames[digit2],"-",
   Print ColorNames[multiplier]
   ```

9. Notice the use of brackets to symbolize indexing into an array variable (`ColorNames`). Test run this new version for various input values and verify for correctness. Save your program.

10. Now on to the final requirement v... To depict the resistor graphically with distinguishable color bands, we have to consider the color for the resistor itself. It cannot be black, or white, or any other standard color code color or we would not be able to see bands of that color. Let’s choose a peach-like color because this would be peachy 😊. In RobotBASIC, summon its handy-dandy Color dialog from the Help menu then click the “Define Custom Colors” button. Click around in the color box and try to find a nice, peachy color. It should something like R/G/B = 250,230,210. Note that you can also type 8-bit values (0..255) into the R/G/B boxes directly. Add the following initialization line at the top of your program with the other initializer lines. Notice how this uses RobotBASIC’s RGB() function to “compute” a color value from red/green/blue components.

   ```plaintext
   ResColor = RGB(250,230,210)
   ```

11. You can now add code to the bottom of your program to draw a peach resistor “body” with leads extending from the ends using the `Rectangle` command previously used. Refer to the Helpfile for details on the `Rectangle` command as well as the `RectangleWH` command, which may also be useful here. Use the above `ResColor` variable for both the pen and fill colors.

12. To draw the color bands, we can again use `RectangleWH` for each band, but we need a slick way of specifying the colors. Let’s add another data array “`ColorCodes`” of numeric color values after `ColorNames`. First check out the Helpfile section “RobotBASIC Constants” and notice that 16 color values are defined as symbolic names – let’s use them!

   ```plaintext
   Data ColorCodes;Black,Brown,Red,???,Yellow,Green,Blue,???,DarkGray,White
   ```

13. Uh oh, what about the two “???”s? Orange and violet are not defined! No problem, find RGB values for these two colors and plug in RGB expressions in place of the ??? as you did in step 10.

14. Now we can draw color band Rectangles in the correct colors using the array indexing trick again:

   ```plaintext
   RectangleWH 400,50,10,30,ColorCodes[digit1],ColorCodes[digit1]
   ```

15. To separate white bands from the terminal window’s background, you should draw your resistor graphic in a non-white area, say medium gray. For example:
16. Complete your programming masterpiece, verify for various test values, and demonstrate your solution to the instructor for completion credit of this lab. Make sure you have a saved copy of your final version.

**Part 2: A Smarter Robot**

Note: this part will likely become a homework assignment.