APS 105 — Computer Fundamentals

Lab #2: Computation and Control Flow in C
Winter, 2018

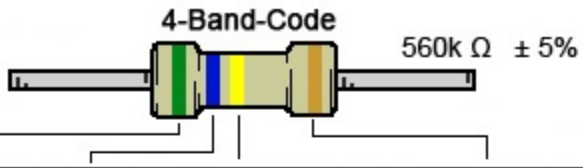
Due date: at the end of your assigned lab session

Objective

In this lab you will write a program to compute the resistance value of a given resistor based on its colour bands, as input by the user.

Resistor Colour Codes

Resistors are painted with coloured bands that can be decoded to determine their resistance value. In this lab, you will be decoding the resistance of 4-band resistors. Figure 1 contains a chart, which is a mapping from the colour of each band to a numerical value. When these 4 values are combined, one can arrive at a resistance value and a tolerance for that resistor.



CHAR	COLOR	1ST BAND	2ND BAND	MULTIPLIER	TOLERANCE
K, k	Black	0	0	1Ω	
B, b	Brown	1	1	10Ω	± 1%
R, r	Red	2	2	100Ω	± 2%
O, o	Orange	3	3	1KΩ	
E, e	Yellow	4	4	10KΩ	
G, g	Green	5	5	100KΩ	± 0.5%
U, u	Blue	6	6	1MΩ	± 0.25%
V, v	Violet	7	7	10MΩ	± 0.10%
Y, y	Grey	8	8		± 0.05%
W, w	White	9	9		
L, l	Gold			0.1Ω	± 5%
S, s	Silver			0.01Ω	± 10%

Figure 1: 4-Band Resistor Colour Code Chart

Given the example in the chart (the colour bands Green, Blue, Yellow, and Gold), the calculation is performed as follows. The first two bands are combined to create a 2-digit number, in this case 56 (Green = 5 and Blue = 6). This value is then multiplied by the value corresponding to the third color, in this case 10 kΩ (Yellow), leaving us with a resistance value of 560 kΩ. Next, since resistors are not perfect, we want to know the uncertainty in the resistance value, which is also known as the tolerance. The fourth and final band contains this value. Finally, after looking up the colour gold in the tolerance column, we arrive at a value of 560 kΩ ± 5%.

Resistance Calculator Program

Write a program, Lab2.c, that starts by prompting the user for the four colour codes of a given resistor. Each colour will be entered by the user as a single character, according to the first column of the chart in Figure 1. Your program will then print out the full names of the four colour bands and the corresponding resistance value.

Sample Output

Example 1:

```
Please enter the 1st band:
g
Please enter the 2nd band:
u
Please enter the multiplier band:
e
Please enter the tolerance band:
l
Resistor bands: Green Blue Yellow Gold
Resistance: 560.00 KOhms +/- 5.00%
```

Example 2:

```
Please enter the 1st band:
K
Please enter the 2nd band:
W
Please enter the multiplier band:
B
Please enter the tolerance band:
Y
Resistor bands: Black White Brown Grey
Resistance: 90.00 Ohms +/- 0.05%
```

Example 3:

```
Please enter the 1st band:
o
Please enter the 2nd band:
v
Please enter the multiplier band:
g
Please enter the tolerance band:
s
Resistor bands: Orange Violet Green Silver
Resistance: 3.70 MOhms +/- 10.00%
```

Notes & Tips

Note that the programs outputs the resistance in units of MOhms (when resistance is greater than or equal to 1 million Ohms), KOhms (when resistance is greater than or equal to 1 thousand Ohms, but less than 1 MOhm), or just in Ohms (when less than 1 KOhm).

Also, when scanning in the characters representing the colour bands, please use scanf as follows (note the space in front of %c):

```
scanf(‘‘ %c’’, &band1);
```

Grading by TA and Submitting for Program for Auto-Marking

You *must* put your C program in a file named `Lab2.c` (be sure to make the capitalization in the file name correct). You can set the name of the file when you first create the project (as described in the CodeLite document) or change the name of the **main.c** file in CodeLite by right clicking on the name, and selecting **Rename...**

There are a total of **10 marks** available in this lab, marked in two different ways:

1. **By your TA, for 4 marks out of 10**. Once you are ready, show your program to your TA so that we can mark your program for style, and to ask you a few questions to test your understanding of what is happening. Programs with good style have been described in class, but briefly, they are:

   - Clear comments that describe what is happening in the program.
   - Good choices for variable names that indicate their purpose (e.g. **inputNumber1** or **divisionOutput** in the case of this lab). Please adopt the following naming convention illustrated above — if have you a variable that is described by multiple words, user lower case for the first letter of the first word, and Upper case for all subsequent words — e.g. **longestLengthString** or **closeWindowDelay**.
   - Properly indented code that has appropriate spacing between lines for readability.

   The TA will also ask you some questions to be sure that you understand the underlying concepts being exercised in this lab.

2. **By an auto-marking program for 6 marks out of 10**. You must sumbit your `Lab2.c` program file through the ECF computers for marking. We will use a software program to compile and run your program, and test it with different inputs. Long before you submit your program for marking, you should run the **exercise** program that compiles and runs your program and gives it sample inputs, and checks that the outputs are correct. To do this you should do the following:

   - Open a terminal window on ECF, using xterm, as described in the Getting Started with Linux and ECF document from Lab 0.
   - Change into the directory (folder) containing your `Lab2.c` file, using the Linux `cd` command.
   - Run the following program by typing:
     `/share/copy/aps105s/lab2/exercise`
     This program will look for the file Lab2.c in your directory, compile it, and run it on a *some* of the test cases that will be used to mark your program automatically later. If there is anything wrong, the exercise program will report this to you, so read its output carefully.

     A key part of what you are to learn in this lab is the use of this program — most software programs give this kind of report. Read through the output of the **exercise** program and see if it is happy with everything, or if it is reporting an error. If there is an error, it will say so, and then it is up to you to fix what is wrong and try again. You'll need to do this for every subsequent lab.

3. Once you have determined that your program is as correct as you can make it, then you must submit your program for auto-marking. This must be done by the end of your lab period (either Friday January 26 or Monday January 29) as that is the due time. To do so, go into the directory containing your `Lab2.c` file (and make sure this is the right place!) and type the following command:
   `/share/copy/aps105s/lab2/submit`
   This command will re-run the exercise program to check that everything looks OK. If it finds a problem, it will ask you if you are sure that you want to submit. Note that you may submit your work as many times as you want prior to the deadline; only the most recent submission is marked.

Important Note: You must submit your lab by the end of your assigned lab period. Late submissions will not be accepted, and you will receive a grade of zero.
You can also check to see if what you think you have submitted is actually there, for peace of mind, using the following command:
`/share/copy/aps105s/lab2/viewsubmitted`
This command will download into the directory you run it in, a copy of all of the files that have been submitted. If you already have files of that same name in your directory, these files will be renamed with a number added to the end of the filename.

After the Final Deadline — Obtaining Automark

After all lab sections have finished (the final one is Monday from 12-2pm), a short time later, you will be able to run the automarker to determine the automarked fraction of your grade on the code you have submitted. To do so run the following command:
`/share/copy/aps105s/lab2/marker`
This command will compile and run your code, and test it with all of the test cases used to determine the automark grade. You will be able to see those test cases' output and what went right or wrong.