EGR 226: Microcontroller Programming and Applications

Winter 2021

Instructor: Prof. Trevor Ekin

Lab 2: Semester Preparation Part 2

Lab Report

Rachel Jacquay

February 3, 2021

Contents

List of Figures

List of Tables

List of Codes

1. Objectives

The objectives of this lab are to practice C programming in preparation for embedded C programming using the TI-MSP432 microcontroller, to practice looping calculation applications using a resistance calculator while taking user input, and to develop a searchable database of books using C-structures.

2. Equipment

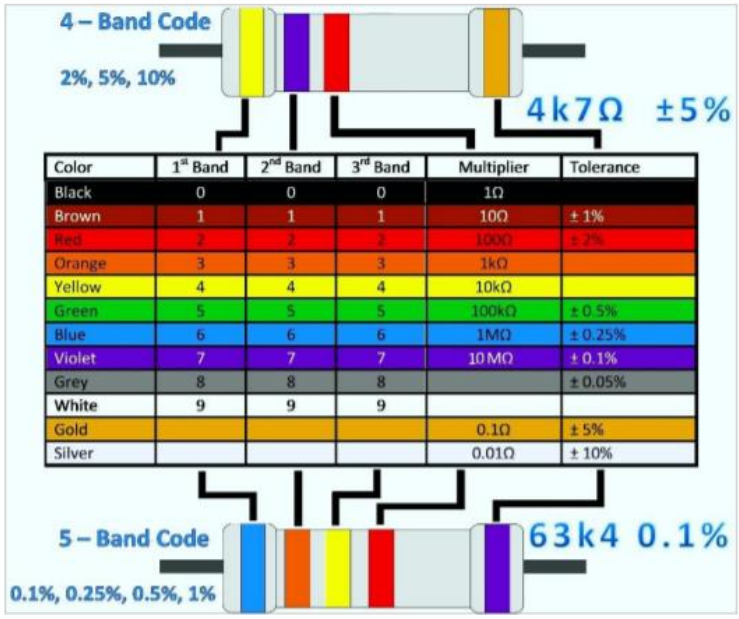
Table 1: Laboratory Equipment Usage

Part	Description	Model	Measured Value	Notes
Code::Blocks IDE	Open-source, cross-platform and free C/C++ IDE	Version 20.03	N / A	N / A

3. Introduction

3.1. Part 1: Resistor Analysis Tool II

In Lab 1, a resistor analysis tool was created that would take a resistance value and output the color-code for that resistance. In Lab 2, this analysis tool was updated to allow the user the choice between entering a resistance or a color-code, outputting the conversion either way. Resistors are an important component used with electronic circuits. One must first become comfortable with the color-code used to determine the amount of resistance, in Ohms (Ω), that a resistor has. Figure 1 below shows the Resistor Color-Code and how it works with resistors of 4- and 5-band codes. This figure also gives an example for each resistor, so that viewers can learn how to use the band codes correctly.



Color	1st Band	2nd Band	3rd Band	Multiplier	Tolerance
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1%
Red	2	2	2	100Ω	± 2%
Orange	3	3	3	1kΩ	
Yellow	4	4	4	10kΩ	
Green	5	5	5	100kΩ	± 0.5%
Blue	6	6	6	1MΩ	± 0.25%
Violet	7	7	7	10MΩ	± 0.1%
Grey	8	8	8		± 0.05%
White	9	9	9		
Gold				0.1Ω	± 5%
Silver				0.01Ω	± 10%

Figure 1: Resistor Color-Code

3.2. Part 2: Book Database

Structures are a very useful tool use to group variables with different or similar data types to organize code. The goal of this part of the lab was to use structures to create a database of books from an external input file. A file, named “BookList.csv,” was read into the program, parsed into an array of “book” structures, then made available for the user to navigate through based their inputs of integers and characters.

4. Procedure

4.1 Part 1: Resistor Analysis Tool II

4.1.1 Steps

The Code::Blocks program was opened, and a new console application was started. Four function prototypes were created at the top of the new file, namely prompt(), getColorBands(), calcResistance(), and calcResistorcolors(). The first function displayed Table 2, shown below, and asked the user if they wanted to decode color-codes into resistances or resistances into color-codes. If the user entered ‘1’, the code ran through the second function, which stored the color-codes, and then the third function, which calculated the resistance from the band colors. If the user entered ‘2’, the fourth function calculated the color-codes of the resistor based on the resistance input from the user. Error check was implemented each time that the user entered a value to see if they were entering the correct

numbers or not. If the user entered '3', the entire program would end, serving as a way to successfully exit the code. The main() function called the prompt() function to request user input for decoding, collected user input for which method of conversion desired, performed the requested task, then immediately looped back to the beginning of itself to see if the user wanted to do any of the requested tasks again.

Table 2: Resistor Color-Code Character Scheme

Character	Color	1st & 2nd Band	3rd Band	4th Band
K	Black	0	*1	+/- 1%
N	Brown	1	*10	+/- 2%
R	Red	2	*100	
O	Orange	3	*1,000	
Y	Yellow	4	*10,000	
G	Green	5	*100,000	+/- 0.5%
B	Blue	6	*1,000,000	+/- 0.25%
V	Violet	7	*10,000,000	+/-0.1%
E	Grey	8		+/- 0.05%
W	White	9		
D	Gold		*0.1	+/- 5%
S	Silver		*0.01	+/- 10%

4.1.2 Code Snippets

In Part 1 of the lab, it was requested as a deliverable that the code has a prompt to tell the user what their options are for the program.

Code 1: Prompt

```
void prompt(void) {     // prompt function definition
    printf("\n");
    printf("---------------------Resistor Codes----------------------\n");
    printf("|Character| Color  | Band 1 & 2 |   Band 3   |  Band 4  |\n");
    printf("|    K    | Black  |      0     |*1          |+/- 1%%    |\n");
    printf("|    N    | Brown  |      1     |*10         |+/- 2%%    |\n");
    printf("|    R    | Red    |      2     |*100        |          |\n");
    printf("|    O    | Orange |      3     |*1,000      |          |\n");
    printf("|    Y    | Yellow |      4     |*10,000     |          |\n");
    printf("|    G    | Green  |      5     |*100,000    |+/- 0.5%%  |\n");
    printf("|    B    | Blue   |      6     |*1,000,000  |+/- 0.25%% |\n");
    printf("|    V    | Violet |      7     |*10,000,000 |+/- 0.1%%  |\n");
    printf("|    E    | Grey   |      8     |            |+/- 0.05%% |\n");
    printf("|    W    | White  |      9     |            |          |\n");
    printf("|    D    | Gold   |            |*0.1        |+/- 5%%    |\n");
    printf("|    S    | Silver |            |*0.01       |+/- 10%%   |\n");
    printf("-------------------------------------------------------\n\n");

    printf("Would you like to convert color-code to resistance or convert resistance to color-code?\n");
    printf("Input '1' for color-code to resistance or '2' for resistance to color-code:\n");
    printf("Input '3' to end program\n\n");
}
```

Code 1 shows how the function prompt() prints out everything that the user needs to know, and tells them to enter 1, 2, or 3.

In Part 1 of the lab, it was requested as a deliverable that the code has a function called calcResistorColors() to take in the resistance from the user and turn it into its respective color-code.

Code 2: Calculate Colors

```
136 void calcResistorColors(int resistance) {        // calculation function definition
137     int i = 0;                                   // counter variable
138     int b1, b2, b3;                              // color bands
139     char color[10][10] = { "Black", "Brown", "Red", "Orange", "Yellow", "Green", "Blue", "Violet", "Grey",
"White" };        // colors
140
141     if (resistance >= 100) {            // if value is greater than 100
142         do {                            // do this while value > 100
143             resistance /= 10;           // divide resistance by 10
144             i++;                        // increment counter by 1
145         } while (resistance >= 100);    // do this while value > 100
146     }
147
148         b1 = resistance / 10;           // band 1 is just resistance / 10
149         b2 = resistance % 10;           // band 2 is the remainder of resistance / 10
150
151         printf("%s-%s-%s\n\n", color[b1], color[b2], color[i]);     // print colors
152     }
```

Code 2 shows how the function calcResistorColors() takes in the resistance from main(), then calculates the colors of the resistor using algebra operations, arrays, printf(), and do-while loops. On line 139, an array is used to create the colors in the color-codes. The calculations are performed on lines 143, 148, and 149.

In Part 1 of the lab, it was requested as a deliverable that the code has a function called getColorBands() to take in the color-code from the user.

Code 3: Get Color Bands

```
161 void getColorBands(char *b1, char *b2, char *b3, char *b4) {        // get color bands from user function
definition
162         int status;
163         int i;
164         int check;
165
166         printf("\nWhich colors should be decoded?\n");
167
168         // letter 1
169         printf("Enter 1st letter: ");
170         do {
171             fflush(stdin);                      // clear std input window
172             status = scanf("%c", b1);           // scan for b1
173
174             if (status == 0) {                  // error check
175                 printf("\nInvalid letter\n");
176                 printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, E, W,
D, or S\n\n");
177                 fflush(stdin);
178             }
179
180         } while(status == 0);   // while the character is not valid
```

Code 3 shows how the function getColorBands() takes in the color-code from main() and error checks to see if the wrong input was entered. The color is being scanned in on line 172. An error check if-statement is on line 174.

In Part 1 of the lab, it was requested as a deliverable that the code has a function called calcResistance() to turn the color-code into a resistance.

Code 4: Calculate Resistance

```c
void calcResistance(char color1, char color2, char color3, char color4) {   // calculate resistance function definition
    int one, two;       // declaring variables
    int check1 = 1;
    int check2 = 1;
    int check3 = 1;
    int check4 = 1;
    float sum = 0;

    // first band
    do {                        // do this while check1 does not equal 0
        switch(color1) {        // switch statement

        case 'K'  :             // both upper and lower case
        case 'k'  :
            one = 0;            // first band
            check1 = 0;         // exit do while loop
            break;


        case 'N'  :             // repeat for all valid letters
        case 'n'  :
            one = 1 * 10;
            check1 = 0;
            break;

        case 'R'  :
        case 'r'  :
            one = 2 * 10;
            check1 = 0;
            break;

        case 'O'  :
        case 'o'  :
            one = 3 * 10;
            check1 = 0;
            break;

        case 'Y'  :
        case 'y'  :
            one = 4;
            check1 = 0;
            break;

        case 'G'  :
        case 'g'  :
            one = 5 * 10;
            check1 = 0;
            break;

        case 'B'  :
        case 'b'  :
            one = 6 * 10;
            check1 = 0;
            break;

        case 'V'  :
        case 'v'  :
            one = 7 * 10;
            check1 = 0;
            break;

        case 'E'  :
        case 'e'  :
            one = 8 * 10;
            check1 = 0;
            break;

        case 'W'  :
        case 'w'  :
            one = 9 * 10;
            check1 = 0;
            break;

        default   :
            printf("\nInput 1 is incorrect\n");     // if user inputs anything other than valid letters, it will stay in default
            errorCheck();                            // telling user that they have entered an invalid letter
            check1 = 0;                              // set check1 equal to 0 to jump out
            break;
        }
    } while (check1 != 0);                           // do while check1 != 0
```

Code 4 shows how the function calcResistance() takes in the letters from getColorBands() and uses switch statements to determine if the values are valid or not, then use those values to make a resistance and then print it out to the terminal window. If the input is not correct, the switch statement will stay in the default case and call the errorCheck() function to let the user know that they need to redo their inputs.

In Part 1 of the lab, it was requested as a deliverable that the program loops until the user wants to exit.

Code 5: Loop

```
35      do {            // do everything until user wants to exit program
36
37      prompt();       // prompt function call
38
39      status == 0;
40      fflush(stdin);
41
42          do {                            // do this to scan in which option user wants
43              fflush(stdin);
44              status = scanf("%d", &onetwo);
45
46              if (status == 0 || onetwo < 1 || onetwo > 3) {
47                  printf("\nInvalid number\n");
48                  printf("Please enter a value either '1' or '2' or '3'\n\n");
49                  fflush(stdin);
50              }
51
52          } while (status == 0 || onetwo < 1 || onetwo > 3);      // do while variable onetwo is not a
variable, is greater than 3 or less than 1
53
54
55      printf("\nUser has entered %d\n", onetwo);      // tell user what they entered
56
57      if (onetwo == 1) {                                      // option 1
58              getColorBands(&b1, &b2, &b3, &b4);              // get color code function call
59
60              printf("\n%c %c %c %c\n", b1, b2, b3, b4);      // print colors from user
61
62              color1 = b1;        // exchanging values since calcResistance takes in char and not char*
63              color2 = b2;
64              color3 = b3;
65              color4 = b4;


66
67              calcResistance(color1, color2, color3, color4); // calculate resistance from colors function
call
68      }
69
70      else if (onetwo == 2) {                                         // option 2
71              fflush(stdin);
72              printf("\nWhat value of resistance should be color-coded?\n");
73              printf("Input a number between 1 and 99,000,000\n");
74              printf("Then press 'Enter'\n\n");
75
76              do {                                    // determine if value entered is valid
77                  status = scanf("%d", &resistance);
78
79                  if (status == 0 || resistance < 1 || resistance > 99000000) {       // if scan is
unsuccessful, or value < 1 or > 99000000
80                      printf("\nInvalid number\n");
81                      printf("Please enter a value between 1 and 99000000\n\n");
82                      fflush(stdin);
83                  }
84
85              } while(status == 0 || resistance < 1 || resistance > 99000000);        // do this for
unsuccessful value, or value < 1 or > 99000000
86
87              printf("\nValid input of %d Ohms\n", resistance);
88              printf("Resistor of %d Ohms would have a color code of:\n\n", resistance);
89
90              calcResistorColors(resistance);     // calculate resistor colors function call
91      }
92
93      else if (onetwo == 3) {     // exit code completely
94          printf("Goodbye!\n");   // say goodbye
95          loop = 0;               // loop is 0 exits code
96      }
97
98      } while (loop != 0);        // repeat entire code until loop equals 0
99  }
```

Code 5 shows how the main() includes a giant do-while loop to make sure that the program runs until loop is set equal to zero, ending the code. There are if-else statements in lines 57, 70, and 90 determining which go to the different parts of the code depending on which number (1, 2, 3) is entered by the user. If the variable equals 3, the loop variable is set to zero, and the code ends. Until then, the whole program continues.

4.2 Part 2: Book Database

4.2.1 Steps

The Code::Blocks program was opened, and a new console application was started, just like in Part 1. In main.c, outside of main(), a structure called “book” was created that contained char title[255], char author_name[50], char ISBN[10], int pages, and int year_published. Each of these are subcategories of the struct. In main(), a 360-element array of book-structures was created. A function called parse_file() was created that takes a parameter of filename, and an array of book structures to populate. This function was called in main() to populate the array with books. A function called print_book() was created that takes a book as a parameter. The function neatly prints out the contents of the matching book’s structure to the output window. A function called search_title() was created that takes a book array, the number of books in that array, and a character string title as parameters. The search function loops through all the books in the array in search of a title that matches with the passes character string, printing to the output window any books that fulfill the requirements. A function called search_author() was created that takes a book array, the number of books in that array, and a character string author as parameters. The search function loops through all the books in the array in search of an author (first and last name) that matches with the passed character string, printing to the output window any books that fulfill the requirements. Finally, a function called search_ISBN() was created that takes a book array, the number of books in that array, and a character string ISBN as parameters. The search function loops through all the books in the array in search of an ISBN that matches with the ISBN passed, printing to the output window any books that fulfill the requirements. In main(), a while-loop monitors user input for an integer that will act as a “search by” selection to offer search features. Error check was implemented each time that the user entered a value to see if they were entering the correct numbers and letters or not.

4.2.2 Code Snippets

In Part 2 of the lab, it was requested as a deliverable that the file be read in and then parsed into words so that the search engine can run correctly.

Code 6: Parse File

```c
int parse_file(char filename[], book book_array[]) {
    char buffer[512];                                   // Create temporary string buffer variable
    int i = 0;
    char* ptr;
    FILE *infile;                                       // Attempt to open file

    infile = fopen(filename, "r");

    if (infile == NULL)                                 // Return 0 (failure) if file could not open
        return -1;

    while (fgets(buffer, 512, infile)) {                // Loop collecting each line from the file
        ptr = strtok(buffer, ",");                      // Parse string by commas and newline
            strcpy(book_array[i].title, ptr);           // First parse is title

        ptr = strtok(NULL, ",\n");                      // repeat for author name
```

```
131             strcpy(book_array[i].author_name, ptr);
132
133         ptr = strtok(NULL, ",\n");                          // repeat for ISBN
134             strcpy(book_array[i].ISBN, ptr);
135
136         ptr = strtok(NULL, ",\n");                          // repeat for pages
137         if (strcmp(ptr, "N/A"))                             // if N/A, set ptr to an int and store the
value in ptr
138             book_array[i].pages = atoi(ptr);
139         else if (strcmp(ptr, "N/A") == 0)                   // if it is 0 to start, output 0
140             book_array[i].pages = 0;
141
142         ptr = strtok(NULL,",\n");                           // repeat for year published
143         if (strcmp(ptr,"N/A"))                              // same as pages
144             book_array[i].year_published = atoi(ptr);
145         else if (strcmp(ptr, "N/A") == 0)
146             book_array[i].year_published = 0;
147
148         i++;        // increment i to see how many books are found
149     }
150
151     fclose(infile);     // close file
152
153     return i;       // return how many books are found total
154 }
```

Code 6 shows how the function parse_file() reads in the file, in line 121, then separates the file's pieces into titles, authors, ISBN's, pages, and years published. There's even an error check to see if the piece in the file is "N/A". On line 151, the file is closed. On line 153, the number of books in the file is returned to main().

In Part 2 of the lab, it was requested as a deliverable that the books, once found as a match, be printed to the terminal window for the user to see.

Code 7: Print Book

```
162 void print_book(book my_book) {                         // printing book function
163     printf("\nTitle:        %s\n", my_book.title);      // print title
164     printf("Author:       %s\n", my_book.author_name);  // print author name
165     printf("ISBN:         %s\n", my_book.ISBN);         // print ISBN
166
167     if (my_book.pages == 0)                             // if struct is 0
168         printf("Pages:        N/A\n");                  // print N/A
169
170     else if (my_book.pages != 0)                        // if struct is not 0
171         printf("Pages:        %d\n", my_book.pages);    // print value
172
173     if (my_book.year_published == 0)                    //if struct is 0
174         printf("Year Published:       N/A\n");          // print N/A
175
176     else if (my_book.year_published != 0)                           // if struct is not 0
177         printf("Year Published:       %d\n", my_book.year_published);   // print value
178 }
```

Code 7 shows how the function print_book() outputs the books to the screen. Line 167 shows the if-else statements where the pages piece being scanned in could possibly be "N/A". Line 173 does the same thing for the year published piece.

In Part 2 of the lab, it was requested as a deliverable that the user can search for a title.

Code 8: Search Title

```
186 void search_title(book book_title[], int n, char title[]) {     // search title function definition
187     int i;
188     int var;
189     char outcome;
190
191     for (i = 0; i <= n; i++) {                          // until index equals n number of books
192         outcome = (strstr(book_title[i].title, title));     // set outcome equal to strstr of book_title[i]
and title
193
194         if (outcome) {                                  // if outcome == 1

195             print_book(book_title[i]);                      // print title
196             var++;                                          // var tells user how many books were found
197             }
198         }
199
200     if (var == 0) {                         // if var == 0
201         printf("\nNo results found\n");     // no books found
202     }
203 }
```

Code 8 shows how the function search_title searches the given string “title” in the main string “book_title” and returns the first occurrence. Line 192 shows is where this takes place with strstr(). If strstr() is not zero, the print_book() function is called and the book is printed. If strstr() is zero, the user is told that there are no results.

In Part 2 of the lab, it was requested as a deliverable that the user can search for an author.

Code 9: Search Author

```
211 void search_author(book book_author[], int n, char author_name[]) {     // search author function
definitions
212     int i;
213     int var;
214     char outcome;
215
216     for (i = 0; i <= n; i++) {                                  // until index is equal to n number
of books
217         outcome = (strstr(book_author[i].author_name, author_name));    // set outcome equal to strstr of
book_author[i] and author
218
219         if (outcome) {                      // if outcome == 1
220             print_book(book_author[i]);     // print author
221             var++;                          // var tells user how many books were found
222         }
223     }
224
225     if (var == 0) {                         // if var == 0
226         printf("\nNo results found\n");     // no books found
227     }
228 }
```

Code 9 shows how the function search_author searches the given string “author” in the main string “book_author” and returns the first occurrence. Line 217 shows is where this takes place with strstr(). If strstr() is not zero, the print_book() function is called and the book is printed. If strstr() is zero, the user is told that there are no results.

In Part 2 of the lab, it was requested as a deliverable that the user can search for an ISBN.

Code 10: Search ISBN

```
236 void search_ISBN(book book_ISBN[], int n, char ISBN[]) {        // search ISBN function definition
237     int i;
238     int var;
239     char outcome;
240
241     for (i = 0; i <= n; i++) {                          // until index is equal to n number of books
242         outcome = (strstr(book_ISBN[i].ISBN, ISBN));    // set outcome equal to strstr of book_ISBN[i] and
ISBN
243
244         if (outcome) {                  // if outcome == 1
245             print_book(book_ISBN[i]);   // print ISBN
246             var++;                      // var tells user how many books were found
247         }
248     }
249
250     if (var == 0) {                         // if var == 0
251         printf("\nNo results found\n");     // no books found
252     }
253 }
```

Code 10 shows how the function search_ISBN searches the given string “ISBN” in the main string “book_ISBN” and returns the first occurrence. Line 242 shows is where this takes place with strstr(). If strstr() is not zero, the print_book() function is called and the book is printed. If strstr() is zero, the user is told that there are no results.

In Part 2 of the lab, it was requested as a deliverable that the code will loop until the user wants to exit.

Code 11: Loop Part 2

```
49      do {                                                    // do all of this while the user still
wants to loop
50          printf("Which process would you like to search by?\n");     // prompt user
51          printf("Please enter one of the following numbers:\n");
52          printf("[0] Search by Title\n");
53          printf("[1] Search by Author\n");
54          printf("[2] Search by ISBN\n");
55          printf("[3] Exit code\n\n");
56
57          do {                                // do all of this while the scanf and numbers scanned in are
valid
58              fflush(stdin);
59              status = scanf("%d", &num);     // checking for scanf to be valid
60
61              if (status == 0 || num < 0 || num > 3) {        // if invalid, redo
62                  printf("Incorrect value\n");
63                  printf("Please enter a '0' or '1' or '2' or '3'\n");
64                  fflush(stdin);
```

```
65              }
66          } while (status == 0 || num < 0 || num > 3);
67
68          if (num == 0) {                     // search by title
69              printf("\nUser entered 0\n");
70
71                  fflush(stdin);
72                  printf("\nPlease enter a case-sensitive title\n\n");
73                  scanf("%[^\n]s", userin);                       // get the string
74                  search_title(book_array, b, userin);        // call search by title function
75          }
76
77          else if (num == 1) {                // search by author
78              printf("\nUser entered 1\n");
79
80                  fflush(stdin);
81                  printf("\nPlease enter a case-sensitive author\n\n");
82                  scanf("%[^\n]s", userin);                       // get the string
83                  search_author(book_array, b, userin);       // call search by author function
84          }
85
86          else if (num == 2) {                // search by ISBN
87              printf("\nUser entered 2\n");
88
89                  fflush(stdin);
90                  printf("\nPlease enter an ISBN\n\n");
91                  scanf("%[^\n]s", userin);                       // get the string
92                  search_ISBN(book_array, b, userin);         // call search by ISBN function
93          }
94
95          else if (num == 3) {                // end the program entirely
96              printf("\nUser entered 3\n");
97              printf("Goodbye!");
98              loop = 0;
99          }
100
101         printf("\n");
102
103     } while (loop != 0);    // loop until user wants to end program
104
105     return 0;
106 }
```

Code 11 shows how the main() function loops until the user tells it to stop. Unless loop equals zero, the main function will continue. The do-while loop starts on line 49 and ends on line 103. Everything within main() in inside the do-while loop, so everything gets redone. The if-else statements on lines 68, 77, 86, and 95 are where the code splits when the user inputs 0, 1, 2, or 3. If the user inputs 0, 1, or 2, the respective function is called and the search engine runs. If the user inputs 3, then loop becomes zero and the code ends.

5. Results / Discussion

5.1 Part 1 Results:

Running the application developed in Part 1 successfully prints the Prompt, as displayed in Figure 2.

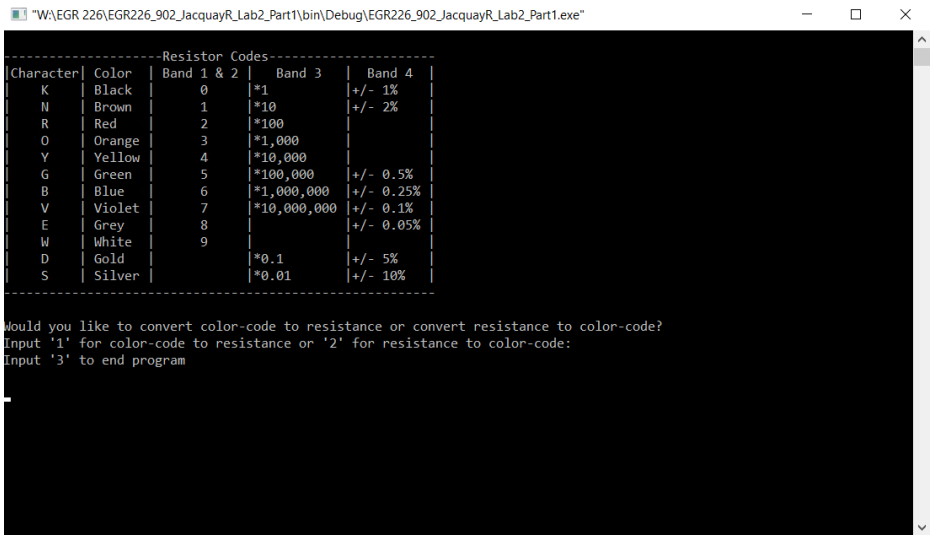


Figure 2: Part 1 Output (1)

After the user enters ‘1’, the program runs Part 1 of the code, which is where it decodes color-bands into a resistance in Ohms, including a tolerance. If any of these inputs are incorrect, the program will reset, and the prompt will be displayed again. Figure 3 shows this.

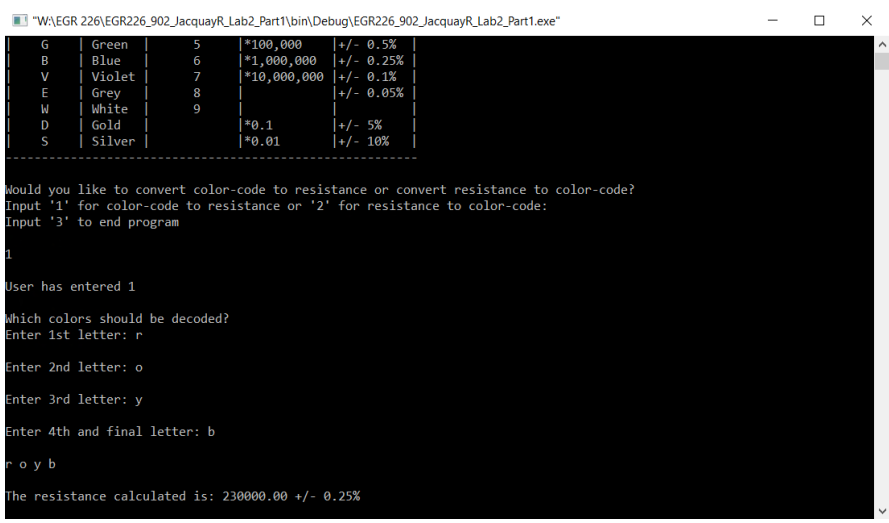


Figure 3: Part 1 Output (2)

After the user enters ‘2’, the program runs Part 2 of the code, which is where it decodes resistance into color-bands. If the input is incorrect, the program will reset, and the prompt will be displayed again. Figure 4 shows this.

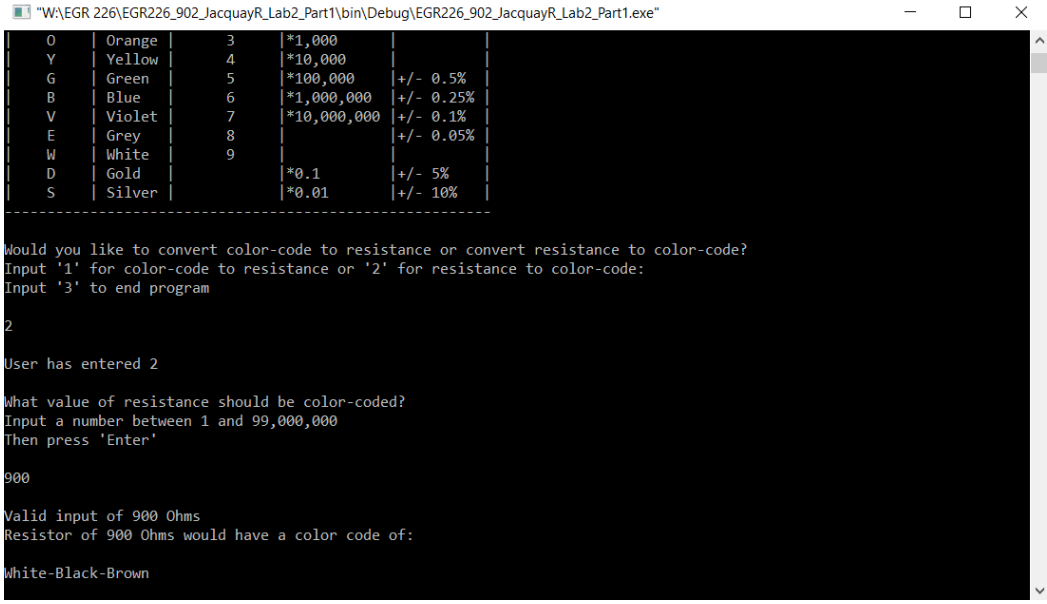


Figure 4: Part 1 Output (3)

After the resistance has been displayed, the program asks if the user would like to input another value, shown in Figure 5. They must enter ‘0’ for ‘No’ and anything else for ‘Yes.’ Figure 5 shows this.

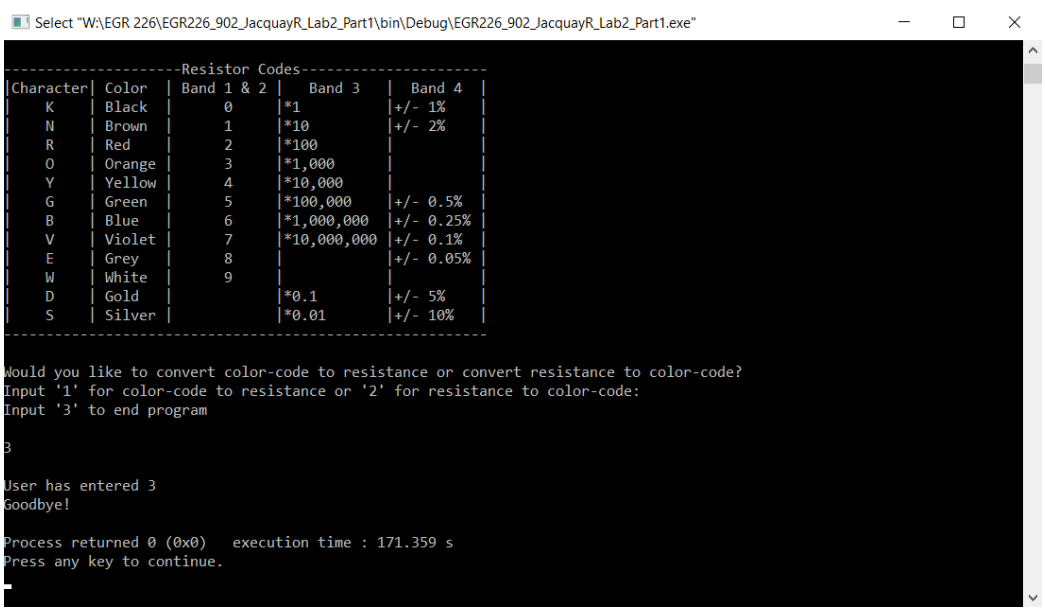


Figure 5: Part 1 Output (4)

5.2 Part 2 Results:

Running the application developed in Part 2 successfully prints the prompt, shown in Figure 7.

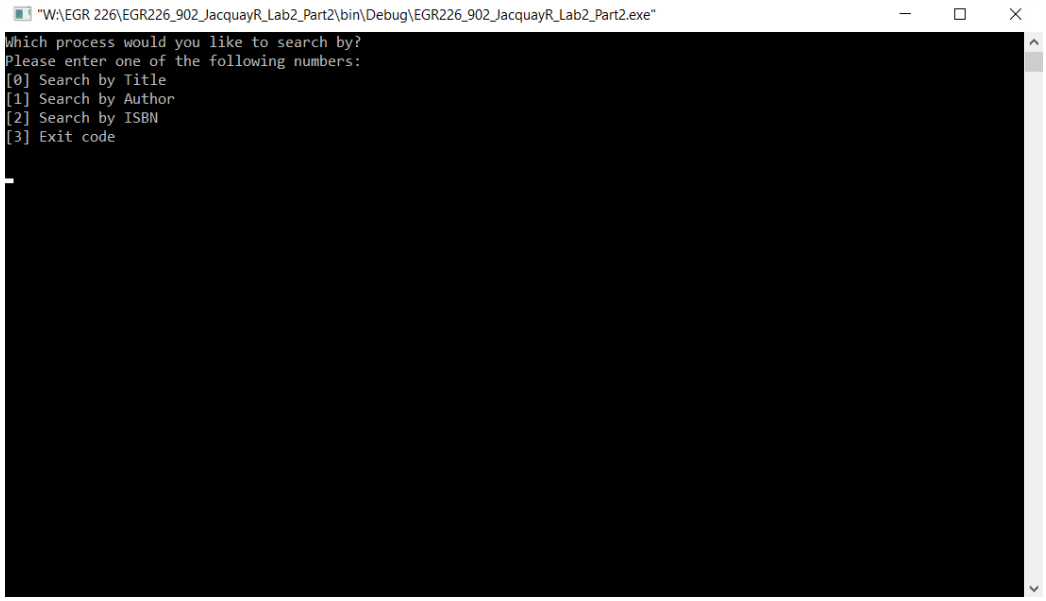


Figure 7: Part 2 Output (1)

After entering '0', the code runs the Title search engine and outputs any book that has a matching word in its title, shown in Figure 8.

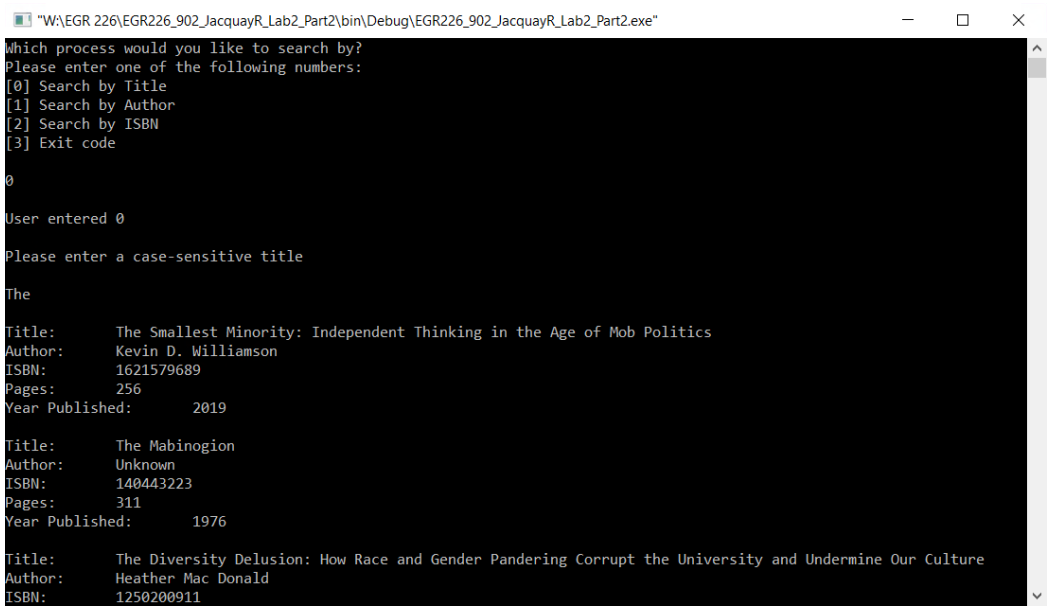


Figure 8: Part 2 Output (2)

After entering ‘1’, the code runs the Author search engine and outputs any book that has a matching word in its author name, shown in Figure 9.

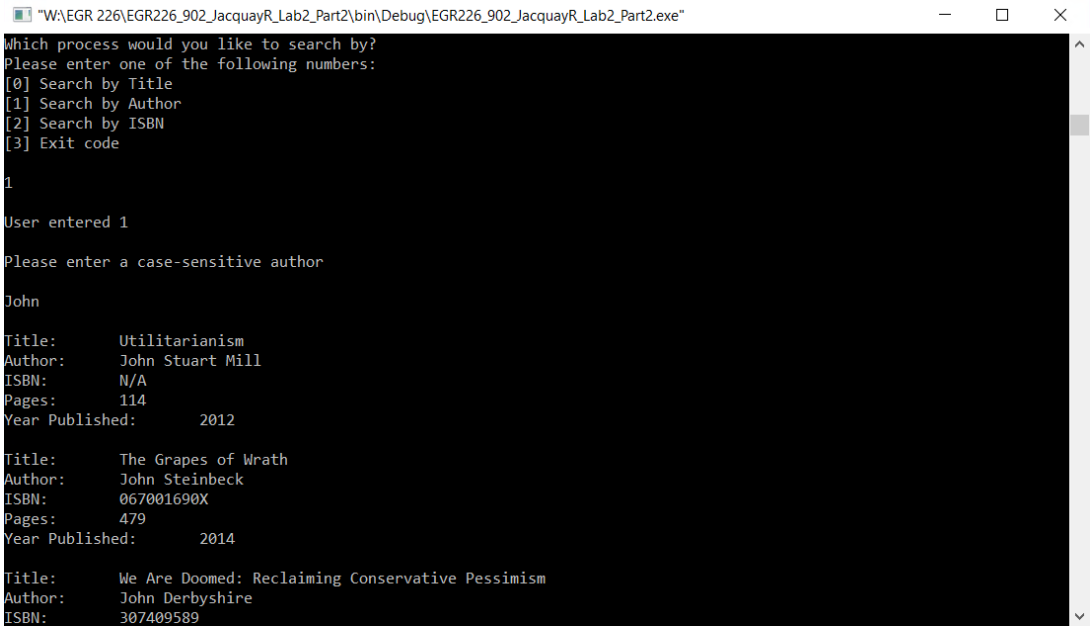


Figure 9: Part 2 Output (3)

After entering ‘2’, the code runs the ISBN search engine and outputs any book that has a matching number in its ISBN, shown in Figure 10.

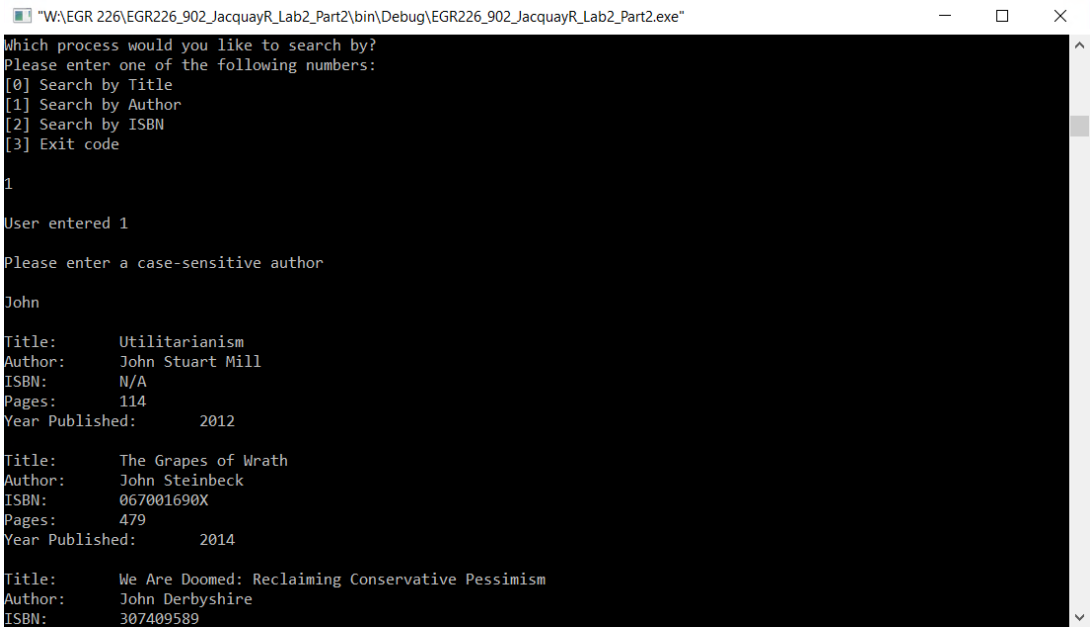


Figure 10: Part 2 Output (4)

After entering ‘3’, the code exits the program entirely, shown in Figure 11.

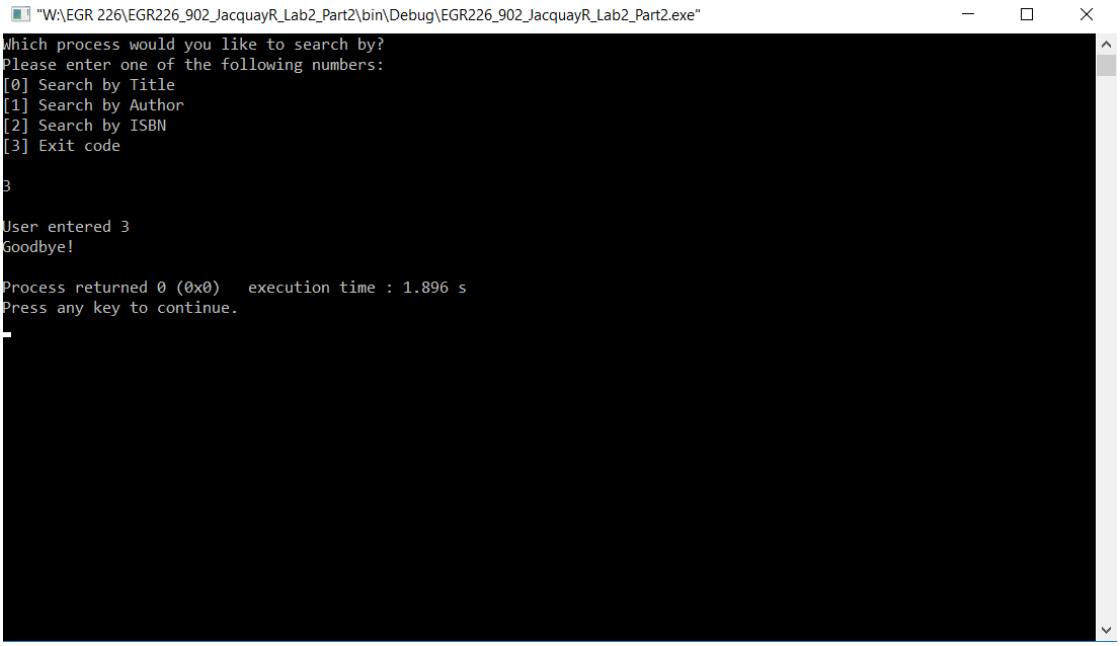


Figure 11: Part 2 Output (5)

6. Conclusion and Future Work

In conclusion, various tools for C programming were implemented during this lab, including loops, arrays, files, pointers, libraries, functions, etc. These allowed for the review of the language before more complicated assignments arise. In Part 1 of this lab, functions were used to display a prompt to the user, scan in their input and store it, convert that value from a resistance to a color-code for a resistor, or vice versa, then output it to the window so that they user could see. In Part 2 of this lab, functions were used to prompt the user at the beginning, as well as to implement the search engine and print out the books that matched. Furthermore, pointers and arrays were used to send values to and from functions in order to create a search engine where the user could search by title, author, or ISBN. Both parts even looped back to the beginning and redid the programs as many times as the user would like.

Some challenges faced while creating this code had to do with pointers, switch statements, and functions. Having not worked with pointers since the fall of 2019, this was tricky. The hardest part was getting the pointers to be sent through functions and then return the values that were stored in them, instead of their addresses. This took some time getting used to, and lots of questions and searches. From personal experience, switch statements have never been a big component of coding in past classes. They were rarely covered and taught. This required a lot of research to make sure that they were set up and executed properly, with the default case being the error case. Switch statements made more sense to use than arrays though, and that’s why they were used in the search engine code. Finally, some problems arose with functions because, at one point, main() was being called from a function, and because main() was looping through, it made the prompt print and loop continue after the user chose to exit. Overall, this was a good review with C, and it required a lot of processing and research. There are no suggestions about implementing better technique or fixing a problem with the code.

Appendices

Source Code: C Review Project

A. main_part1.c

```c
/*****************************************************************************
* Author:       Rachel Jacquay
* Course:       EGR 226-902
* Date:         01/27/2021
* Project:      Lab 2 Part 1
* File:         main_part1.c
* Description:  This program has two parts. One part takes in the color code of a
                hypothetical resistor, then computes and outputs the corresponding
                resistance. The other part takes in the resistance code of a
                hypothetical resistor, then decodes it into its corresponding
                color codes. It loops until the user tells it to stop by setting the
                variable 'loop' to 0. Error checking was used for all inputs by the
                user.
*****************************************************************************/

#include <stdio.h>      // preprocessor directives
#include <stdlib.h>
#include <math.h>

void prompt(void);              // function prototypes
void calcResistorColors(int);
void getColorBands(char*, char*, char*, char*);
void errorCheck(void);
void calcResistance(char, char, char, char);

int main()                  // main function
{
    int resistance;                         // the user's inputed resistance value
    int status = 0;                         // to check if the scan was successful
    int loop = 1;                           // keep user in loop to enter more resistance
    int onetwo;                             // pick which calculation is occurring
    char *b1, *b2, *b3, *b4;                // pointers to letters
    char color1, color2, color3, color4;    // letters from b1, b2, b3, b4

    do {            // do everything until user wants to exit program

    prompt();       // prompt function call

    status == 0;
    fflush(stdin);

        do {                                // do this to scan in which option user wants
            fflush(stdin);
            status = scanf("%d", &onetwo);

            if (status == 0 || onetwo < 1 || onetwo > 3) {
                printf("\nInvalid number\n");
                printf("Please enter a value either '1' or '2' or '3'\n\n");
                fflush(stdin);
            }

        } while (status == 0 || onetwo < 1 || onetwo > 3);      // do while variable onetwo is not a variable, is greater than 3 or less than 1


    printf("\nUser has entered %d\n", onetwo);      // tell user what they entered

    if (onetwo == 1) {                                  // option 1
            getColorBands(&b1, &b2, &b3, &b4);          // get color code function call

            printf("\n%c %c %c %c\n", b1, b2, b3, b4);      // print colors from user

            color1 = b1;        // exchanging values since calcResistance takes in char and not char*
            color2 = b2;
            color3 = b3;
            color4 = b4;
```

```c

            calcResistance(color1, color2, color3, color4); // calculate resistance from colors function call
    }

    else if (onetwo == 2) {                                                 // option 2
            fflush(stdin);
            printf("\nWhat value of resistance should be color-coded?\n");
            printf("Input a number between 1 and 99,000,000\n");
            printf("Then press 'Enter'\n\n");

            do {                                    // determine if value entered is valid
                status = scanf("%d", &resistance);

                if (status == 0 || resistance < 1 || resistance > 99000000) {       // if scan is unsuccessful, or value < 1 or > 99000000
                    printf("\nInvalid number\n");
                    printf("Please enter a value between 1 and 99000000\n\n");
                    fflush(stdin);
                }

            } while(status == 0 || resistance < 1 || resistance > 99000000);        // do this for unsuccessful value, or value < 1 or > 99000000

            printf("\nValid input of %d Ohms\n", resistance);
            printf("Resistor of %d Ohms would have a color code of:\n\n", resistance);

            calcResistorColors(resistance);     // calculate resistor colors function call
    }

    else if (onetwo == 3) {     // exit code completely
        printf("Goodbye!\n");   // say goodbye
        loop = 0;               // loop is 0 exits code
    }

    } while (loop != 0);        // repeat entire code until loop equals 0
}

/*  prompt
    Description: This function allows for the user to be shown the color code
    table and explained the idea of this code. They're asked to input 1, 2, or 3.
    Inputs: none
    Outputs: none
*/
void prompt(void) {     // prompt function definition
    printf("\n");
    printf("--------------------Resistor Codes---------------------\n");
    printf("|Character| Color  | Band 1 & 2 |   Band 3   |  Band 4  |\n");
    printf("|    K    | Black  |     0      |*1          |+/- 1%%    |\n");
    printf("|    N    | Brown  |     1      |*10         |+/- 2%%    |\n");
    printf("|    R    | Red    |     2      |*100        |          |\n");
    printf("|    O    | Orange |     3      |*1,000      |          |\n");
    printf("|    Y    | Yellow |     4      |*10,000     |          |\n");
    printf("|    G    | Green  |     5      |*100,000    |+/- 0.5%%  |\n");
    printf("|    B    | Blue   |     6      |*1,000,000  |+/- 0.25%% |\n");
    printf("|    V    | Violet |     7      |*10,000,000 |+/- 0.1%%  |\n");
    printf("|    E    | Grey   |     8      |            |+/- 0.05%% |\n");
    printf("|    W    | White  |     9      |            |          |\n");
    printf("|    D    | Gold   |            |*0.1        |+/- 5%%    |\n");
    printf("|    S    | Silver |            |*0.01       |+/- 10%%   |\n");
    printf("-------------------------------------------------------\n\n");

    printf("Would you like to convert color-code to resistance or convert resistance to color-code?\n");
    printf("Input '1' for color-code to resistance or '2' for resistance to color-code:\n");
    printf("Input '3' to end program\n\n");
}
```

```c

/*  calcResistorColors
    Description: This function allows for the resistances given by the user
    to be decoded into the color bands of the resistor.
    Inputs: resistance
    Outputs: none
*/
void calcResistorColors(int resistance) {        // calculation function definition
    int i = 0;                                   // counter variable
    int b1, b2, b3;                              // color bands
    char color[10][10] = { "Black", "Brown", "Red", "Orange", "Yellow", "Green", "Blue", "Violet", "Grey",
"White" };        // colors

    if (resistance >= 100) {            // if value is greater than 100
        do {                            // do this while value > 100
            resistance /= 10;           // divide resistance by 10
            i++;                        // increment counter by 1
        } while (resistance >= 100);    // do this while value > 100
    }

        b1 = resistance / 10;           // band 1 is just resistance / 10
        b2 = resistance % 10;           // band 2 is the remainder of resistance / 10

        printf("%s-%s-%s\n\n", color[b1], color[b2], color[i]);     // print colors
    }

/*  getColorBands
    Description: This function allows for the colors to be collected
    from the user and stored in pointers to be brought back to the
    main function:
    Inputs: b1, b2, b3, b4
    Outputs: none
*/
void getColorBands(char *b1, char *b2, char *b3, char *b4) {        // get color bands from user function
definition
        int status;
        int i;
        int check;

        printf("\nWhich colors should be decoded?\n");

        // letter 1
        printf("Enter 1st letter: ");
        do {
            fflush(stdin);                       // clear std input window
            status = scanf("%c", b1);            // scan for b1

            if (status == 0) {                   // error check
                printf("\nInvalid letter\n");
                printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, E, W,
D, or S\n\n");
                fflush(stdin);
            }

        } while(status == 0);   // while the character is not valid

        // letter 2
        printf("\nEnter 2nd letter: ");     // do the same for letter 2
        do {
            fflush(stdin);
            status = scanf("%c", b2);

            if (status == 0) {
                printf("\nInvalid letter\n");
                printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, E, or
W\n\n");
```

```
                fflush(stdin);

            }

        } while(status == 0);


        // letter 3
        printf("\nEnter 3rd letter: ");     // do the same for letter 3
        do {
            fflush(stdin);
            status = scanf("%c", b3);

            if (status == 0) {
                printf("\nInvalid letter\n");
                printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, D, or
S\n\n");
                fflush(stdin);

            }

        } while(status == 0);


        // letter 4
        printf("\nEnter 4th and final letter: ");       // do the same for letter 4
        do {
            fflush(stdin);
            status = scanf("%c", b4);

            if (status == 0) {
                printf("\nInvalid letter\n");
                printf("Please enter one of the following uppercase letters: K, N, G, B, V, E, D, or S\n\n"
);
                fflush(stdin);

            }

        } while(status == 0);
}

/*  calcResistance
    Description: This function allows for the colors given by the user to
    be decoded into their colors.
    Inputs: color1, color2, color3, color4
    Outputs: none
*/
void calcResistance(char color1, char color2, char color3, char color4) {   // calculate resistance
function definition
    int one, two;       // declaring variables
    int check1 = 1;
    int check2 = 1;
    int check3 = 1;
    int check4 = 1;
    float sum = 0;

    // first band
    do {                        // do this while check1 does not equal 0
        switch(color1) {        // switch statement

        case 'K'  :             // both upper and lower case
        case 'k'  :
            one = 0;            // first band
            check1 = 0;         // exit do while loop
            break;

```

```
        case 'N'  :             // repeat for all valid letters
        case 'n'  :
            one = 1 * 10;
            check1 = 0;
            break;

        case 'R'  :
        case 'r'  :
            one = 2 * 10;
            check1 = 0;
            break;

        case 'O'  :
        case 'o'  :
            one = 3 * 10;
            check1 = 0;
            break;

        case 'Y'  :
        case 'y'  :
            one = 4;
            check1 = 0;
            break;

        case 'G'  :
        case 'g'  :
            one = 5 * 10;
            check1 = 0;
            break;

        case 'B'  :
        case 'b'  :
            one = 6 * 10;
            check1 = 0;
            break;

        case 'V'  :
        case 'v'  :
            one = 7 * 10;
            check1 = 0;
            break;

        case 'E'  :
        case 'e'  :
            one = 8 * 10;
            check1 = 0;
            break;

        case 'W'  :
        case 'w'  :
            one = 9 * 10;
            check1 = 0;
            break;

        default   :
            printf("\nInput 1 is incorrect\n");     // if user inputs anything other than valid letters, it will stay in default
            errorCheck();                            // telling user that they have entered an invalid letter
            check1 = 0;                              // set check1 equal to 0 to jump out
            break;
        }
    } while (check1 != 0);                           // do while check1 != 0

    sum += one;                 // add the value to sum

```

```
    // second band
    do {
        switch(color2) {            // repeat everything in the first switch statement for band 2

        case 'K'  :
        case 'k'  :
            two = 0;
            check2 = 0;
            break;

        case 'N'  :
        case 'n'  :
            two = 1;
            check2 = 0;
            break;

        case 'R'  :
        case 'r'  :
            two = 2;
            check2 = 0;
            break;

        case 'O'  :
        case 'o'  :
            two = 3;
            check2 = 0;
            break;

        case 'Y'  :
        case 'y'  :
            two = 4;
            check2 = 0;
            break;

        case 'G'  :
        case 'g'  :
            two = 5;
            check2 = 0;
            break;

        case 'B'  :
        case 'b'  :
            two = 6;
            check2 = 0;
            break;

        case 'V'  :
        case 'v'  :
            two = 7;
            check2 = 0;
            break;

        case 'E'  :
        case 'e'  :
            two = 8;
            check2 = 0;
            break;

        case 'W'  :
        case 'w'  :
            two = 9;
            check2 = 0;
            break;

        default   :
            printf("\nInput 2 is incorrect\n");
```

```
            errorCheck();
            check2 = 0;
            break;
        }
    } while (check2 != 0);

    sum += two;             // add the value to sum to print out resistance later

    // third band
    do {
        switch(color3) {            // repeat everything in the first switch statement for band 3

        case 'K'  :
        case 'k'  :
            sum *= 1;               // sum is multiplied by a constant now
            check3 = 0;
            break;

        case 'N'  :
        case 'n'  :
            sum *= 10;
            check3 = 0;
            break;

        case 'R'  :
        case 'r'  :
            sum *= 100;
            check3 = 0;
            break;

        case 'O'  :
        case 'o'  :
            sum *= 1000;
            check3 = 0;
            break;

        case 'Y'  :
        case 'y'  :
            sum *= 10000;
            check3 = 0;
            break;

        case 'G'  :
        case 'g'  :
            sum *= 100000;
            check3 = 0;
            break;

        case 'B'  :
        case 'b'  :
            sum *= 1000000;
            check3 = 0;
            break;

        case 'V'  :
        case 'v'  :
            sum *= 10000000;
            check3 = 0;
            break;

        case 'D'  :
        case 'd'  :
            sum *= 0.1;
            check3 = 0;
            break;

```

```
        case 'S'  :
        case 's'  :
            sum *= 0.01;
            check3 = 0;
            break;

        default   :
            printf("\nInput 3 is incorrect\n");
            errorCheck();
            check3 = 0;
            break;
        }
    } while (check3 != 0);

    printf("\nThe resistance calculated is: %.2f", sum);    // output what the resistance calculated is

    // fourth band
    do {
        switch(color4) {            // repeat everything in the first switch statement for band 4

        case 'K'  :
        case 'k'  :
            printf(" +/- 1%%");     // just print out what the tolerances are
            check4 = 0;
            break;

        case 'N'  :
        case 'n'  :
            printf(" +/- 2%%");
            check4 = 0;
            break;

        case 'G'  :
        case 'g'  :
            printf(" +/- 0.5%%");
            check4 = 0;
            break;

        case 'B'  :
        case 'b'  :
            printf(" +/- 0.25%%");
            check4 = 0;
            break;

        case 'V'  :
        case 'v'  :
            printf(" +/- 0.1%%");
            check4 = 0;
            break;

        case 'E'  :
        case 'e'  :
            printf(" +/- 0.05%%");
            check4 = 0;

        case 'D'  :
        case 'd'  :
            printf(" +/- 5%%");
            check4 = 0;
            break;

        case 'S'  :
        case 's'  :
            printf(" +/- 10%%");
            check4 = 0;
            break;
```

```c

        default   :
            printf("\nInput 4 is incorrect\n");
            errorCheck();
            check4 = 0;
            break;
        }
    } while (check4 != 0);

    printf("\n");
}

/*  errorCheck
    Description: This function allows for the color code inputs to be
    flashed and redone. The code only gets to this file if the user
    inputs a number or a letter other than the ones required.
    Inputs: none
    Outputs: none
*/
void errorCheck(void) {                         // error check comes straight from default case when
user inputs invalid value
    printf("\nUser entered an invalid value\n");
    printf("Program is reset\n");
}
```

B. main_part2.c

```c
/*****************************************************************************
* Author:       Rachel Jacquay
* Course:       EGR 226-902
* Date:         01/27/2021
* Project:      Lab 2 Part 2
* File:         main_part2.c
* Description:  This program takes in a file of type .csv, and also user input to
                create a search engine of books, using their title, author, ISBN,
                page number, and year published. It loops until the user tells it to
                stop by setting the variable 'loop' equal to 0. Error checking was
                used for all inputs by the user as well as the file.
*****************************************************************************/

#include <stdio.h>      // preprocessor directives
#include <stdlib.h>
#include <math.h>
#include <ctype.h>
#include <string.h>

#define MAX 500         // macro

typedef struct {                // creating struct
        char title[225];
        char author_name[50];
        char ISBN[10];
        int pages;
        int year_published;
    } book;

int parse_file(char filename[], book book_array[]);             // function prototypes
void print_book(book my_book);
void search_title(book book_title[], int n, char title[]);
void search_author(book book_author[], int n, char author_name[]);
void search_ISBN(book book_ISBN[], int n, char ISBN[]);

int main() {                // main function
    book my_book;           // declaring variables
    book book_array[360];
    char filename[MAX];
    char userin[255];
    int b, num;
    int status = 1;
    int loop = 1;

    strcpy(filename, "BookList.csv");       // set filename to BookList.csv

    b = parse_file(filename, book_array);       // set i equal to the number of books read in

    do {                                                    // do all of this while the user still wants to loop
        printf("Which process would you like to search by?\n");     // prompt user
        printf("Please enter one of the following numbers:\n");
        printf("[0] Search by Title\n");
        printf("[1] Search by Author\n");
        printf("[2] Search by ISBN\n");
        printf("[3] Exit code\n\n");

        do {                                // do all of this while the scanf and numbers scanned in are valid
            fflush(stdin);
            status = scanf("%d", &num);     // checking for scanf to be valid

            if (status == 0 || num < 0 || num > 3) {        // if invalid, redo
                printf("Incorrect value\n");
                printf("Please enter a '0' or '1' or '2' or '3'\n");
                fflush(stdin);
```

```
            }
        } while (status == 0 || num < 0 || num > 3);

        if (num == 0) {                     // search by title
            printf("\nUser entered 0\n");

                fflush(stdin);
                printf("\nPlease enter a case-sensitive title\n\n");
                scanf("%[^\n]s", userin);                       // get the string
                search_title(book_array, b, userin);        // call search by title function
        }

        else if (num == 1) {                // search by author
            printf("\nUser entered 1\n");

                fflush(stdin);
                printf("\nPlease enter a case-sensitive author\n\n");
                scanf("%[^\n]s", userin);                       // get the string
                search_author(book_array, b, userin);       // call search by author function
        }

        else if (num == 2) {                // search by ISBN
            printf("\nUser entered 2\n");

                fflush(stdin);
                printf("\nPlease enter an ISBN\n\n");
                scanf("%[^\n]s", userin);                       // get the string
                search_ISBN(book_array, b, userin);         // call search by ISBN function
        }

        else if (num == 3) {                // end the program entirely
            printf("\nUser entered 3\n");
            printf("Goodbye!");
            loop = 0;
        }

        printf("\n");

    } while (loop != 0);    // loop until user wants to end program

    return 0;
}

/*  parse_file
    Description: This function opens, stores, and closes the file, while
    also parsing its contents into words and using those for the search
    engine. It returns the total number of books found.
    Inputs: filename[], book_array[]
    Outputs: i
*/
int parse_file(char filename[], book book_array[]) {
    char buffer[512];                               // Create temporary string buffer variable
    int i = 0;
    char* ptr;
    FILE *infile;                                   // Attempt to open file

    infile = fopen(filename, "r");

    if (infile == NULL)                             // Return 0 (failure) if file could not open
        return -1;

    while (fgets(buffer, 512, infile)) {            // Loop collecting each line from the file
        ptr = strtok(buffer, ",");                  // Parse string by commas and newline
            strcpy(book_array[i].title, ptr);       // First parse is title

        ptr = strtok(NULL, ",\n");                  // repeat for author name
```

```
            strcpy(book_array[i].author_name, ptr);

        ptr = strtok(NULL, ",\n");                          // repeat for ISBN
            strcpy(book_array[i].ISBN, ptr);

        ptr = strtok(NULL, ",\n");                          // repeat for pages
        if (strcmp(ptr, "N/A"))                             // if N/A, set ptr to an int and store the value in ptr
            book_array[i].pages = atoi(ptr);
        else if (strcmp(ptr, "N/A") == 0)                   // if it is 0 to start, output 0
            book_array[i].pages = 0;

        ptr = strtok(NULL,",\n");                          // repeat for year published
        if (strcmp(ptr,"N/A"))                             // same as pages
            book_array[i].year_published = atoi(ptr);
        else if (strcmp(ptr, "N/A") == 0)
            book_array[i].year_published = 0;

        i++;        // increment i to see how many books are found
    }

    fclose(infile);     // close file

    return i;       // return how many books are found total
}

/*  print_book
    Description: This function prints out the book info whenever it is called
    in any of other functions.
    Inputs: my_book
    Outputs: none
*/
void print_book(book my_book) {                         // printing book function
    printf("\nTitle:        %s\n", my_book.title);      // print title
    printf("Author:       %s\n", my_book.author_name);  // print author name
    printf("ISBN:         %s\n", my_book.ISBN);         // print ISBN

    if (my_book.pages == 0)                             // if struct is 0
        printf("Pages:        N/A\n");                  // print N/A

    else if (my_book.pages != 0)                        // if struct is not 0
        printf("Pages:        %d\n", my_book.pages);    // print value

    if (my_book.year_published == 0)                    //if struct is 0
        printf("Year Published:       N/A\n");          // print N/A

    else if (my_book.year_published != 0)                           // if struct is not 0
        printf("Year Published:       %d\n", my_book.year_published);   // print value
}

/*  search_title
    Description: This function determines if the title given by the user
    matches any title in the .csv file.
    Inputs: book_title, n, title
    Outputs: none
*/
void search_title(book book_title[], int n, char title[]) {     // search title function definition
    int i;
    int var;
    char outcome;

    for (i = 0; i <= n; i++) {                          // until index equals n number of books
        outcome = (strstr(book_title[i].title, title));     // set outcome equal to strstr of book_title[i] and title

        if (outcome) {                                  // if outcome == 1
```

```
            print_book(book_title[i]);                      // print title
            var++;                                          // var tells user how many books were found
            }
        }

    if (var == 0) {                         // if var == 0
        printf("\nNo results found\n");     // no books found
    }
}

/*  search_author
    Description: This function determines if the author name given by the user
    matches any author names in the .csv file.
    Inputs: book_author, n, author_name
    Outputs: none
*/
void search_author(book book_author[], int n, char author_name[]) {     // search author function
definitions
    int i;
    int var;
    char outcome;

    for (i = 0; i <= n; i++) {                                          // until index is equal to n number
of books
        outcome = (strstr(book_author[i].author_name, author_name));    // set outcome equal to strstr of
book_author[i] and author

        if (outcome) {                      // if outcome == 1
            print_book(book_author[i]);     // print author
            var++;                          // var tells user how many books were found
        }
    }

    if (var == 0) {                         // if var == 0
        printf("\nNo results found\n");     // no books found
    }
}

/*  search_ISBN
    Description: This function determines if the ISBN given by the user
    matches any ISBN in the .csv file.
    Inputs: book_ISBN, n, ISBN
    Outputs: none
*/
void search_ISBN(book book_ISBN[], int n, char ISBN[]) {        // search ISBN function definition
    int i;
    int var;
    char outcome;

    for (i = 0; i <= n; i++) {                          // until index is equal to n number of books
        outcome = (strstr(book_ISBN[i].ISBN, ISBN));    // set outcome equal to strstr of book_ISBN[i] and
ISBN

        if (outcome) {                  // if outcome == 1
            print_book(book_ISBN[i]);   // print ISBN
            var++;                      // var tells user how many books were found
        }
    }

    if (var == 0) {                         // if var == 0
        printf("\nNo results found\n");     // no books found
    }
}
```