

```

1  /*****
2  * Author:      Rachel Jacquay
3  * Course:     EGR 226-902
4  * Date:       01/27/2021
5  * Project:    Lab 2 Part 1
6  * File:       main_part1.c
7  * Description: This program has two parts. One part takes in the color code of a
8                 hypothetical resistor, then computes and outputs the corresponding
9                 resistance. The other part takes in the resistance code of a
10                hypothetical resistor, then decodes it into its corresponding
11                color codes. It loops until the user tells it to stop by setting the
12                variable 'loop' to 0. Error checking was used for all inputs by the
13                user.
14  *****/
15
16  #include <stdio.h>          // preprocessor directives
17  #include <stdlib.h>
18  #include <math.h>
19
20  void prompt(void);          // function prototypes
21  void calcResistorColors(int);
22  void getColorBands(char*, char*, char*, char*);
23  void errorCheck(void);
24  void calcResistance(char, char, char, char);
25
26  int main()                  // main function
27  {
28      int resistance;          // the user's inputted resistance value
29      int status = 0;          // to check if the scan was successful
30      int loop = 1;            // keep user in loop to enter more resistance
31      int onetwo;              // pick which calculation is occurring
32      char *b1, *b2, *b3, *b4; // pointers to letters
33      char color1, color2, color3, color4; // letters from b1, b2, b3, b4
34
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     }
100
101     /* prompt
102     Description: This function allows for the user to be shown the color code
103     table and explained the idea of this code. They're asked to input 1, 2, or 3.
104     Inputs: none
105     Outputs: none
106     */
107     void prompt(void) { // prompt function definition
108         printf("\n");
109         printf("-----Resistor Codes-----\n");
110         printf("|Character| Color | Band 1 & 2 | Band 3 | Band 4 |\n");
111         printf("| K | Black | 0 | *1 | +/- 1% |\n");
112         printf("| N | Brown | 1 | *10 | +/- 2% |\n");
113         printf("| R | Red | 2 | *100 | |\n");
114         printf("| O | Orange | 3 | *1,000 | |\n");
115         printf("| Y | Yellow | 4 | *10,000 | |\n");
116         printf("| G | Green | 5 | *100,000 | +/- 0.5% |\n");
117         printf("| B | Blue | 6 | *1,000,000 | +/- 0.25% |\n");
118         printf("| V | Violet | 7 | *10,000,000 | +/- 0.1% |\n");
119         printf("| E | Grey | 8 | | +/- 0.05% |\n");
120         printf("| W | White | 9 | | |\n");
121         printf("| D | Gold | | *0.1 | +/- 5% |\n");
122         printf("| S | Silver | | *0.01 | +/- 10% |\n");
123         printf("-----\n\n");
124
125         printf("Would you like to convert color-code to resistance or convert resistance to color-code?\n");
126         printf("Input '1' for color-code to resistance or '2' for resistance to color-code:\n");
127         printf("Input '3' to end program\n\n");
128     }

```

```

129
130 /* calcResistorColors
131 Description: This function allows for the resistances given by the user
132 to be decoded into the color bands of the resistor.
133 Inputs: resistance
134 Outputs: none
135 */
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 }
153
154 /* getColorBands
155 Description: This function allows for the colors to be collected
156 from the user and stored in pointers to be brought back to the
157 main function:
158 Inputs: b1, b2, b3, b4
159 Outputs: none
160 */
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
181
182     // letter 2
183     printf("\nEnter 2nd letter: "); // do the same for letter 2
184     do {
185         fflush(stdin);
186         status = scanf("%c", b2);
187
188         if (status == 0) {
189             printf("\nInvalid letter\n");
190             printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, E, or
W\n\n");

```

```

191         fflush(stdin);
192
193     }
194
195     } while(status == 0);
196
197
198     // letter 3
199     printf("\nEnter 3rd letter: "); // do the same for letter 3
200     do {
201         fflush(stdin);
202         status = scanf("%c", b3);
203
204         if (status == 0) {
205             printf("\nInvalid letter\n");
206             printf("Please enter one of the following uppercase letters: K, N, R, O, Y, G, B, V, D, or S\n");
207             fflush(stdin);
208
209         }
210
211     } while(status == 0);
212
213
214     // letter 4
215     printf("\nEnter 4th and final letter: "); // do the same for letter 4
216     do {
217         fflush(stdin);
218         status = scanf("%c", b4);
219
220         if (status == 0) {
221             printf("\nInvalid letter\n");
222             printf("Please enter one of the following uppercase letters: K, N, G, B, V, E, D, or S\n");
223             fflush(stdin);
224
225         }
226
227     } while(status == 0);
228 }
229
230 /* calcResistance
231 Description: This function allows for the colors given by the user to
232 be decoded into their colors.
233 Inputs: color1, color2, color3, color4
234 Outputs: none
235 */
236 void calcResistance(char color1, char color2, char color3, char color4) { // calculate resistance
237     // function definition
238     int one, two; // declaring variables
239     int check1 = 1;
240     int check2 = 1;
241     int check3 = 1;
242     int check4 = 1;
243     float sum = 0;
244
245     // first band
246     do { // do this while check1 does not equal 0
247         switch(color1) { // switch statement
248
249             case 'K' : // both upper and lower case
250             case 'k' :
251                 one = 0; // first band
252                 check1 = 0; // exit do while loop
253                 break;

```

```

254         case 'N' : // repeat for all valid letters
255         case 'n' :
256             one = 1 * 10;
257             check1 = 0;
258             break;
259
260         case 'R' :
261         case 'r' :
262             one = 2 * 10;
263             check1 = 0;
264             break;
265
266         case 'O' :
267         case 'o' :
268             one = 3 * 10;
269             check1 = 0;
270             break;
271
272         case 'Y' :
273         case 'y' :
274             one = 4;
275             check1 = 0;
276             break;
277
278         case 'G' :
279         case 'g' :
280             one = 5 * 10;
281             check1 = 0;
282             break;
283
284         case 'B' :
285         case 'b' :
286             one = 6 * 10;
287             check1 = 0;
288             break;
289
290         case 'V' :
291         case 'v' :
292             one = 7 * 10;
293             check1 = 0;
294             break;
295
296         case 'E' :
297         case 'e' :
298             one = 8 * 10;
299             check1 = 0;
300             break;
301
302         case 'W' :
303         case 'w' :
304             one = 9 * 10;
305             check1 = 0;
306             break;
307
308         default :
309             printf("\nInput 1 is incorrect\n"); // if user inputs anything other than valid letters, it
will stay in default
310             errorCheck(); // telling user that they have entered an invalid
letter
311             check1 = 0; // set check1 equal to 0 to jump out
312             break;
313     }
314 } while (check1 != 0); // do while check1 != 0
315
316 sum += one; // add the value to sum
317

```

```
318 // second band
319 do {
320     switch(color2) { // repeat everything in the first switch statement for band 2
321
322         case 'K' :
323         case 'k' :
324             two = 0;
325             check2 = 0;
326             break;
327
328         case 'N' :
329         case 'n' :
330             two = 1;
331             check2 = 0;
332             break;
333
334         case 'R' :
335         case 'r' :
336             two = 2;
337             check2 = 0;
338             break;
339
340         case 'O' :
341         case 'o' :
342             two = 3;
343             check2 = 0;
344             break;
345
346         case 'Y' :
347         case 'y' :
348             two = 4;
349             check2 = 0;
350             break;
351
352         case 'G' :
353         case 'g' :
354             two = 5;
355             check2 = 0;
356             break;
357
358         case 'B' :
359         case 'b' :
360             two = 6;
361             check2 = 0;
362             break;
363
364         case 'V' :
365         case 'v' :
366             two = 7;
367             check2 = 0;
368             break;
369
370         case 'E' :
371         case 'e' :
372             two = 8;
373             check2 = 0;
374             break;
375
376         case 'W' :
377         case 'w' :
378             two = 9;
379             check2 = 0;
380             break;
381
382         default :
383             printf("\nInput 2 is incorrect\n");
```

```

384         errorCheck();
385         check2 = 0;
386         break;
387     }
388 } while (check2 != 0);
389
390 sum += two;           // add the value to sum to print out resistance later
391
392 // third band
393 do {
394     switch(color3) {           // repeat everything in the first switch statement for band 3
395
396     case 'K' :
397     case 'k' :
398         sum *= 1;           // sum is multiplied by a constant now
399         check3 = 0;
400         break;
401
402     case 'N' :
403     case 'n' :
404         sum *= 10;
405         check3 = 0;
406         break;
407
408     case 'R' :
409     case 'r' :
410         sum *= 100;
411         check3 = 0;
412         break;
413
414     case 'O' :
415     case 'o' :
416         sum *= 1000;
417         check3 = 0;
418         break;
419
420     case 'Y' :
421     case 'y' :
422         sum *= 10000;
423         check3 = 0;
424         break;
425
426     case 'G' :
427     case 'g' :
428         sum *= 100000;
429         check3 = 0;
430         break;
431
432     case 'B' :
433     case 'b' :
434         sum *= 1000000;
435         check3 = 0;
436         break;
437
438     case 'V' :
439     case 'v' :
440         sum *= 10000000;
441         check3 = 0;
442         break;
443
444     case 'D' :
445     case 'd' :
446         sum *= 0.1;
447         check3 = 0;
448         break;
449

```

```

450     case 'S' :
451     case 's' :
452         sum *= 0.01;
453         check3 = 0;
454         break;
455
456     default :
457         printf("\nInput 3 is incorrect\n");
458         errorCheck();
459         check3 = 0;
460         break;
461 }
462 } while (check3 != 0);
463
464 printf("\nThe resistance calculated is: %.2f", sum); // output what the resistance calculated is
465
466 // fourth band
467 do {
468     switch(color4) { // repeat everything in the first switch statement for band 4
469
470     case 'K' :
471     case 'k' :
472         printf(" +/- 1%%"); // just print out what the tolerances are
473         check4 = 0;
474         break;
475
476     case 'N' :
477     case 'n' :
478         printf(" +/- 2%%");
479         check4 = 0;
480         break;
481
482     case 'G' :
483     case 'g' :
484         printf(" +/- 0.5%%");
485         check4 = 0;
486         break;
487
488     case 'B' :
489     case 'b' :
490         printf(" +/- 0.25%%");
491         check4 = 0;
492         break;
493
494     case 'V' :
495     case 'v' :
496         printf(" +/- 0.1%%");
497         check4 = 0;
498         break;
499
500     case 'E' :
501     case 'e' :
502         printf(" +/- 0.05%%");
503         check4 = 0;
504
505     case 'D' :
506     case 'd' :
507         printf(" +/- 5%%");
508         check4 = 0;
509         break;
510
511     case 'S' :
512     case 's' :
513         printf(" +/- 10%%");
514         check4 = 0;
515         break;

```



```

516
517     default    :
518         printf("\nInput 4 is incorrect\n");
519         errorCheck();
520         check4 = 0;
521         break;
522     }
523 } while (check4 != 0);
524
525 printf("\n");
526 }
527
528 /*  errorCheck
529     Description: This function allows for the color code inputs to be
530     flushed and redone. The code only gets to this file if the user
531     inputs a number or a letter other than the ones required.
532     Inputs: none
533     Outputs: none
534 */
535 void errorCheck(void) { // error check comes straight from default case when
user inputs invalid value
536     printf("\nUser entered an invalid value\n");
537     printf("Program is reset\n");
538 }

```