```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;

            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;

```

```c
        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;

        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
    flushed 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");
}
```