```
/*
    Please code solutions for the problems below.  You should only have one main
function
    and call the required functions for each problem.  Sample output:

    Enter the colors of the resistor's three bands, beginning with
    the band nearest the end. Type the colors in lowercase letters
    only, NO CAPS.
    Band 1 => green
    Band 2 => black
    Band 3 => yellow
    Resistance value: 500 kilo-ohm(s)
    Do you want to decode another resistor: y

    Enter the colors of the resistor's three bands, beginning with
    the band nearest the end. Type the colors in lowercase letters
    only, NO CAPS.
    Band 1 => brown
    Band 2 => vilet
    Band 3 => gray
    Resistance value: -1 kilo-ohm(s)
    Do you want to decode another resistor: n

    Enter a string: chair
    The plural of chair is chairs
    Do you want to convert another string: y

    Enter a string: dairy
    The plural of dairy is dairies
    Do you want to convert another string: y

    Enter a string: boss
    The plural of boss is bosses
    Do you want to convert another string: n

    Enter first string: procrastination
    Enter second string: destination
    The suffix of procrastination and destination is stination
    Do you want to find another suffix: y

    Enter first string: globally
    Enter second string: internally
    The suffix of globally and internally is ally
    Do you want to find another suffix: y

    Enter first string: gloves
    Enter second string: dove
    The suffix of gloves and dove is
    Do you want to find another suffix: n


    Process returned 0 (0x0)   execution time : 124.085 s
    Press any key to continue.


*/

// Import standard input/output library
#include <stdio.h>
```

```
#include <string.h>
#include <math.h>

// Function prototypes go here
void resistor(char colorOne[7], char colorTwo[7], char colorThree[7]);
int searchIndex(char colorToSearch[7]);
void reverseString(char stringToReverse[]);
void plural(char noun[]);
void getLastTwoCharacters(char givenString[], char lastTwoCharacters[3]);
void suffix(char wordOne[], char wordTwo[], char commonSuffix[]);
void removeCharacterFromString(char *p, char c);


//global variable
char COLOR_CODES[10][7] = {"black", "brown", "red", "orange", "yellow", "green",
"blue", "violet", "gray", "white"};
int  COLOR_CODE_ARRAY_LENGTH = 10;

// Main function goes here
int main()
{

    char colorBandOne[10];
    char colorBandTwo[10];
    char colorBandThree[10];
    char yesNoOne[] = "y";
    char yesNoTwo[] = "y";
    char yesNoThree[] = "y";

while (strcmp(yesNoOne, "y") == 0)
{
    printf("\nBand 1 => ");
    scanf("%s", colorBandOne);

    printf("\nBand 2 => ");
    scanf("%s", colorBandTwo);

    printf("\nBand 3 => ");
    scanf("%s", colorBandThree);

    //Display resistance value
    resistor(colorBandOne, colorBandTwo, colorBandThree);

    printf("\n Do you want to decode another resistor: ");
    scanf("%s", yesNoOne);

}

//problem 2: Plurals
while (strcmp(yesNoTwo, "y") == 0)
    {
        char userNoun[20];
        printf("Please enter a noun: ");
        scanf("%s", userNoun);
        plural(userNoun);

        printf("\n Do you want to convert another strings: ");
        scanf("%s", yesNoTwo);
    }
//problem 3: longest suffix
while (strcmp(yesNoThree, "y") == 0)
    {
        char commonSuffixString[20];
        char string1[50]; // procrastination, globally, doves
        char string2[50]; // destination, internally, cave

        printf("Please enter a word: ");
        scanf("%s", string1);

        printf("Please enter another word with a similar suffix: ");
        scanf("%s", string2);

        suffix(string1, string2, commonSuffixString);

        printf("\n The longest common suffix of %s and %s is: %s \n", string1,
string2, commonSuffixString);

        printf("\n Do you want to find another suffix: ");
        scanf("%s", yesNoThree);
    }

    return 1;
}


// Main, libraries and prototypes: 5 points



/*

Problem 1: 10 points

A resistor is a circuit device designed to have a specific resistance value
between its ends. Resistance values are expressed in ohms or kilo-ohms.
Resistors are frequently marked with colored bands that encode their
resistance values, as shown below. The first two bands are digits, and the
third is a power-of-ten multiplier.

The table below shows the meanings of each band color. For example, if the
first band is green, the second is black, and the third is yellow, the resistor has
a
value of 50 x 10^4 or 500k.  The 50 is 10 x green(index 5) + black(index 0)).  The
10^4 is 10^yellow(index 4).  The information in the table can be stored in a C
program
as an array of strings.

    char COLOR_CODES[10][7] = {"black", "brown", "red",
    "orange", "yellow", "green", "blue", "violet", "gray",
    "white"};

Notice that "red" is COLOR_CODES[2] and has a digit value of 2 and a multiplier
value of 10 2 . In general, COLOR_CODES[ n ] has digit value n and multiplier value
10 n .
Write a function that accepts the colors of Band 1, Band 2, and Band 3,
and then returns the resistance in kilo-ohms. Include a helper function search
that takes three parameters--the list of strings, the size of the list, and a
target
string, and returns the subscript of the list element that matches the target or
returns -1 if the target is not in the list.

Color Codes for Resistors*
    Color Value     Digit Value     Multiplier
    Black           0               10^0
    Brown           1               10^1
    Red             2               10^2
    Orange          3               10^3
    Yellow          4               10^4
    Green           5               10^5
    Blue            6               10^6
    Violet          7               10^7
    Gray            8               10^8
    White           9               10^9

*Adapted from Sears and Zemanskyís University Physics , 10th edited by Hugh D.
Young and Roger A.
Freedman (Boston: Addison-Wesley, 2000), p. 807.

*/

// resistor function goes here
void resistor(char colorOne[7], char colorTwo[7], char colorThree[7])
{
    //Find the Index of Colors in Color Array
    int indexOfColor1 = searchIndex(colorOne);
    int indexOfColor2 = searchIndex(colorTwo);
    int indexOfColor3 = searchIndex(colorThree);

    //Calculate the resistance value only if all colors are found in the color
array
    double resistance = -1;
    if (indexOfColor1 >= 0 && indexOfColor2 >= 0 && indexOfColor3 >= 0)
    {
        resistance = ((indexOfColor1 * 10) + (indexOfColor2)) * pow(10,
indexOfColor3);
        //Convert to Kilo-Ohms
        resistance /= 1000;
    }

    printf("Resistance Value: %lf kilo-ohm(s)", resistance);

}


// search function goes here
int searchIndex(char colorToSearch[7])
{
    int indexOfColor = -1;
    for (int i = 0; i < COLOR_CODE_ARRAY_LENGTH; i++)
    {
        if (strcmp(COLOR_CODES[i], colorToSearch) == 0)
        {
            indexOfColor = i;
            break;
        }
    }


    return indexOfColor;
}


/*

Problem 2: 5 points

Write a function that takes nouns and forms their plurals on the basis of
these rules:

    a. If noun ends in "y", remove the "y" and add "ies".
    b. If noun ends in "s", "ch", or "sh", add "es".
    c. In all other cases, just add "s".

Print each noun and its plural. Try the following data:
    chair   dairy   boss    circus  fly dog church  clue    dish

*/

// plural function goes here
void plural(char noun[])
{
    int nounLength = strlen(noun);

    if (nounLength > 0)
    {
        int pluralLength = nounLength + 5;
        char plural[pluralLength];
        for(int i = 0; i < pluralLength; i++)
        {
            plural[i] = '\0';
        }

        char endingCharacter = noun[nounLength - 1];

        char lastTwoCharacters[3];
        getLastTwoCharacters(noun, lastTwoCharacters);

        if(endingCharacter == 's' || strcmp(lastTwoCharacters, "ch") == 0 ||
strcmp(lastTwoCharacters, "sh") == 0)
        {
            strcat(noun, "es");
        }
        else if (endingCharacter == 'y')
        {
            removeCharacterFromString(noun, 'y');
            strncpy(plural, noun, nounLength - 1);
            strcat(noun, "ies");

        }
        else
        {
            strcat(noun, "s");
        }
```

```c
        printf("\n The plural is %s", noun);
    }
}

//Get last two characters of a string
void getLastTwoCharacters(char givenString[], char lastTwoCharacters[3])
{

    int givenStringLength = strlen(givenString);
    if (givenStringLength > 2)
    {
        lastTwoCharacters[0] = givenString[givenStringLength - 2];
        lastTwoCharacters[1] = givenString[givenStringLength - 1];
        lastTwoCharacters[2] = '\0';
    }

}

void removeCharacterFromString(char *p, char c)
{
    if (NULL == p)
        return;

    char * pDest = p;

    while (*p)
    {
        if(*p != c)
        {
            *pDest ++ = *p;
        }
        p++;
    }
    *pDest = '\0';
}


/*

Problem 3: 5 points

Write and test a function that finds and returns through an output parameter
the longest common suffix of two words (e.g., the longest common suffix of
"procrastination" and "destination" is "stination", of "globally" and "internally"
is "ally", and of "gloves" and "dove" is the empty string).

*/

// suffix function goes here

void suffix(char wordOne[], char wordTwo[], char commonSuffix[])
{
   int wordOneLength = strlen(wordOne);
   int wordTwoLength = strlen(wordTwo);

   int charactersToCompare = 0;

   if (wordOneLength <= wordTwoLength)
```

```
    {
        charactersToCompare = wordOneLength;
    }
    else
    {
        charactersToCompare = wordTwoLength;
    }

    int indexWordOne = wordOneLength - 1;
    int indexWordTwo = wordTwoLength - 1;
    for (int i = 0; i < charactersToCompare; i++)
    {
        if (wordOne[indexWordOne] == wordTwo[indexWordTwo])
        {
            commonSuffix[i] = wordOne[indexWordOne];
            indexWordOne--;
            indexWordTwo--;
        }
        else
        {
            commonSuffix[i] = '\0';
        }

    }
    int commonSuffixLength = strlen(commonSuffix);
    if(commonSuffixLength > 0)
    {
        reverseString(commonSuffix);
    }

}

//String Reverse Function

 void reverseString(char stringToReverse[])
 {
     int stringLength = strlen(stringToReverse);

     if (stringLength > 1)
     {
         char tempString[stringLength + 1];
         strcpy(tempString, stringToReverse);
         int i = 0;

         for (; i < stringLength; i++)
         {
            stringToReverse[i] = tempString[stringLength - i - 1];

         }

        stringToReverse[i] = '\0';

     }

 }
```