Sample Code in this file: Python, JavaScript, C++, C.

Sample Python Code

```python
# Author : Nezar Assawiel
# Short program to return the total area of two rectangles considering they might interserct

def totalAreaOfTwoRect(xmin1, ymin1, xmax1, ymax1, xmin2, ymin2, xmax2, ymax2):
    ''' Return the total area of two rectangles. If they intersect, intersection
    area counted for only once
    (xmin, ymin) : bottom left corner of rectangle
    (xmax, ymax) : upper right corner of rectangle
    '''


    from collections import namedtuple
    Rectangle = namedtuple('Rectangle', 'xmin ymin xmax ymax')

    a= Rectangle(xmin1, ymin1, xmax1, ymax1)
    b= Rectangle(xmin2, ymin2, xmax2, ymax2)

    totalArea = area (a) + area(b) - intersec(a,b)

    return totalArea


def intersec(a,b):
    '''Return area of intersection between two rectangles. If no intersection,
    return 0.
    Each rectangle is passed as namedtuple('Rectangle', 'xmin ymin xmax ymax')
    '''

    xx = min(a.xmax, b.xmax) - max(a.xmin, b.xmin)
    yy = min(a.ymax, b.ymax) - max(a.ymin, b.ymin)

    if (xx>=0) and (yy>=0):
        return xx*yy


def area (r):
    '''Calculate area of rectangle.rectangle is passed as
    namedtuple('Rectangle', 'xmin ymin xmax ymax')'''

    width = r.xmax - r.xmin
    height= r.ymax - r.ymin

    return width*height
```

Sample JavaScript Code

```
/**--------------------
 Author: Nezar Assawiel
    Date: Dec. 2015
----------------------
*/

/*
The elevator problem! This seems to be a standard coding problem which
asks for the total number of stops an elevator should make given the inputs:
A: Passengers' weights (array)
B: Passengers' destination floor (array)
M: Number of floors
X: Elevator Max Capacity (# of passengers)
Y  Elevator Max Capacity (total weight of passengers)
 */


function solution(A, B, M, X, Y) {

    let dests = [], totalWeight = 0, elevTrip = 0;

    for (let i = 0; i < A.length; i += 1) {

        // check if possible to fit 1 more person in current trip
        if (typeof dests[elevTrip] !== 'undefined') {

            // check if capacity is reached. If such, make a new trip
            if (totalWeight + A[i] > Y || dests[elevTrip].length === X ) {

                elevTrip++; totalWeight = 0;
            }
        }

        //increase current total weight
        totalWeight += A[i];

        //create empty array for destinations
        dests[elevTrip] = dests[elevTrip] || [];

        //push destination to current trip
        dests[elevTrip].push(B[i]);

    }

    // remove duplicate destinations from each trip, since there will be
    //one stop for all of them. (adding 1 is for the return to ground floor)
    dests = dests.map(x => RemoveDup(x).length + 1);

    // return sum of stops in each trip
    return dests.reduce((before, current) => before + current, 0);
}


//remove duplicate elements in array
function RemoveDup(A) {
    return A.reduce((before, current) => {

        if (before.indexOf(current) === -1)
        {before = before.concat(current);
    }
        return before;}, []);
}
```

Sample C++ Code

```
/*

This is a OO program that converts seconds inputted as a number to time in the form hr:min:sec (00:00:00)
and compares the time of two clocks.
The task can be accomplished using simpler programs. But, this is to showcase usage of C++'s OO and string
manipulation
The program is organized into two files: a header files that contains all the definitions and a main file

-----------------------
Author: Nezar Assawiel
Date: 26 Nov 2015.
-----------------------

*/

//-----------------------------------------Header file clockConv.h-----------------------------------------
#ifndef clockConv
#define clockConv
#include <iostream>
using namespace std;



class clockInst
{
        friend ostream& operator<<(ostream&, const clockInst&);
        friend istream& operator>>(istream&, clockInst&);


public:

        // set member variables hr, mins and sec
        void setTime(int hours, int minutes, int seconds);

        //return the time
        void getTime(int& hours, int& minutes, int& seconds) const;

        //Increment time by 1 second
        clockInst operator++();

        //True if time of both clocks are equal, otherwise false
        bool operator==(const clockInst& otherClock) const;

        //True if the time of clock 1 and 2 are different
        bool operator!=(const clockInst& otherClock) const;


        //True if time of clock1 =< clock2
        bool operator<=(const clockInst& otherClock) const;

        //True if time of clock1 < clock2
        bool operator<(const clockInst& otherClock) const;

        //True if time of clock1 >= clock2
        bool operator>=(const clockInst& otherClock) const;

        //True if time of clock1 > clock2
        bool operator>(const clockInst& otherClock) const;

        //Initialize the object with the values supplied by user or use default if no values
        //were supplied
        clockInst(int hours = 0, int minutes = 0, int seconds = 0);


private:
        // Variables to store hours, minutes and seconds
        int hr;
        int min;
        int sec;
};

#endif
```

```
//Return the incremented value of the object
clockInst clockInst::operator++()
{
	sec++;
	if (sec > 59)
	{
		sec = 0;
		min++;
		if (min > 59)
		{
			min = 0;
			hr++;
				if (hr > 23)
				hr = 0;
		}
	}
	return *this;
}


//Overload the (equality) operator.
bool clockInst::operator==(const clockInst& otherClock) const
{
	return (hr == otherClock.hr && min == otherClock.min && sec == otherClock.sec);
}


//Overload the (<=)operator.
bool clockInst::operator<=(const clockInst& otherClock) const
{
return ((hr < otherClock.hr) || (hr == otherClock.hr && min < otherClock.min) ||
		(hr == otherClock.hr && min == otherClock.min && sec <= otherClock.sec));

}


//Overload the (not equal) operator.
bool clockInst::operator!=(const clockInst& otherClock) const
{
return (hr != otherClock.hr || min != otherClock.min || sec != otherClock.sec);
}


//Overload the (<) operator.
bool clockInst::operator<(const clockInst& otherClock) const
{
return ((hr < otherClock.hr) || (hr == otherClock.hr && min < otherClock.min) ||
		(hr == otherClock.hr && min == otherClock.min && sec < otherClock.sec));
}

//Overload the (>=)  operator.
bool clockInst::operator>=(const clockInst& otherClock) const

{
return ((hr > otherClock.hr) || (hr == otherClock.hr && min > otherClock.min) ||
		(hr == otherClock.hr && min == otherClock.min && sec >= otherClock.sec));
}



//Overload the (>)  operator.
bool clockInst::operator>(const clockInst& otherClock) const
{
return ((hr > otherClock.hr) || (hr == otherClock.hr && min > otherClock.min) ||
		(hr == otherClock.hr && min == otherClock.min && sec > otherClock.sec));
}


void clockInst::setTime(int hours, int minutes, int seconds)
{
	if (0 <= hours && hours < 24)
	hr = hours;
	else
	hr = 0;

	if (0 <= minutes && minutes < 60)
	min = minutes;
	else
```

```
        min = 0;

        if (0 <= seconds && seconds < 60)
        sec = seconds;
        else
        sec = 0;
}

void clockInst::getTime(int& hours, int& minutes, int& seconds) const
{
        hours = hr;
        minutes = min;
        seconds = sec;
}


//Constructor
clockInst::clockInst(int hours, int minutes, int seconds)
{
        setTime(hours, minutes, seconds);
}

//Overload the stream insertion operator.
ostream& operator<<(ostream& osObject, const clockInst& timeOut)
{
        if (timeOut.hr < 10)
        osObject << '0';
        osObject << timeOut.hr << ':';
        if (timeOut.min < 10)
        osObject << '0';
        osObject << timeOut.min << ':';
        if (timeOut.sec < 10)
        osObject << '0';
        osObject << timeOut.sec;
        return osObject; //return the ostream object
}


//overload the stream extraction operator
istream& operator>> (istream& is, clockInst& timeIn)
{
        char ch;

        is >> timeIn.hr;

        if (timeIn.hr < 0 || timeIn.hr >= 24)
                timeIn.hr = 0;

        is.get(ch);

        is >> timeIn.min;

        if (timeIn.min < 0 || timeIn.min >= 60)
                timeIn.min = 0;

        is.get(ch);

        is >> timeIn.sec;

        if (timeIn.sec < 0 || timeIn.sec >= 60)
                timeIn.sec = 0;

        return is;
}

//end of header file
//-----------------------------------------------------------------------------------------------------------------


//------------------------------------------------- Main file -----------------------------------------------------

#include <iostream>
#include <string>
#include "clockConv.h"

using namespace std;


int main()
```

```
{
	cout<< "Program to convert and check if converting seconds to the form hr:min:sec is correct "<<endl;


	//define variable
	string UserChoice;


	do {

		//declare insttances of the clock class

		clockInst Clock1;
		clockInst Clock2;
		int NumSeconds;


		//show the content of the variables
		cout << "Clock1(before adding seconds) = " << Clock1 << endl;
		cout << "Clock2 (after adding seconds) = " << Clock2 << endl;

		//ask user to inter time to which the seconds will be addeds
		cout << "Enter the time for Clock1 " << "[hr:min:sec] ";
		cin >> Clock1;


		cout << "Enter the number of seconds to add to Clock1" << "[hr:min:sec] "; //Line 5
		cin >> NumSeconds;
		cout << endl;

		//show user input
		cout << "The time entered for "Clock1" = " << Clock1<<" and for "seconds" = "<<NumSeconds <<
endl;

		Clock2= Clock1;

		//add seconds
		for (int i=0; i<NumSeconds; i++)
		{
			++Clock2;
		}
		cout << "So the new times:" << endl;
		cout << "Clock1 = " << Clock1 << endl;
		cout << "Clock2 = " << Clock2 << endl;


		// comparion statements
		if (Clock1 == Clock2)
			cout << "The times of Clock1 and " << "Clock2 are equal." << endl;


		if (Clock1 < Clock2)
			cout << "The time of Clock1 is less than the time of Clock2." << endl;
		else
			cout << "The time of Clock1 is greater than the time of Clock2." << endl;

		//ask the user if he wishes to run the program again
	    cout << "Do you wish to try again? (Enter y or n for yes or no)" << endl;
	    cin>> UserChoice; cout<<endl;

		while (UserChoice != "y" && UserChoice!="n")
			{
			 cout<<"Invalid choice. Enter y or n" <<endl;
			 cin>>UserChoice; cout<<endl;
			}


	} while ( UserChoice=="y");


	return 0;
}//end of main file ----------------------------------------------------------------------------------------
```

Sample C Code

==========

```
/*
-----------------------
Author: Nezar Assawiel
Date: 20 Sept 2014.
-----------------------

Resistance value of a 3 band resistor (expressed in ohms) can be obtained from the 3 colored bands
 that encode this value .
 The first two bands are digits and the third is a power-of-ten multiplier. For exmaple: if
 the 1st color is yellow, the 2nd is black and the 3rd is orange, the resitor vlaue is:
 (40 x 10^3) or 40 kOhms.

The following table shows the colors and the corresponding values :
Color  Digit_value    Multiplier_value
Black  0                          1
Brown  1                          10
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

*/


#include <stdio.h>
#include <math.h>
#include <ctype.h>
#include <string.h>

//constants
#define SUB_1 10        //number of colors
#define SUB_2 7                 //length of largest string
#define NOT_FOUND -1



int LookFor(const char [][SUB_2], const char [], int);

int main(void)
{
        char user_choice; // "y" to continue program,  otehrwise terminate
        char char_left;   //char left in input stream

        do {

                int i;
                int count;
                int color_index;                    // location of input_color found in list
                double resis_value = 0.0;       // calculated value in kilo-ohms*/
                int no_error = 1;

                // initialize colors array
                char COLOR_CODES[SUB_1][SUB_2] = { "black", "brown", "red", "orange", "yellow", "green", "blue",
                        "violet", "gray", "white"};


                char input_color[SUB_2]; // input_color string array


                printf("Enter the 3 colors of the resistor's bands, starting from the band closeest to the
tail.\n");
                printf("(Note: Tail is the end of the resistor closer to the color bands than the other end)
\n");
                printf("Type colors in lowercase letters (even  1st letter): ");


                for(count = 1; count < 4 && no_error; count++)
                {
                        printf("Band %d => ", count);
                        scanf("%s", input_color);
```

```
                        color_index = LookFor(COLOR_CODES, input_color, SUB_1); // Look for the color


                        if(color_index != NOT_FOUND)
                        {
                                switch(count)
                                {
                                        case 1: resis_value = color_index * 10;
                                        break;


                                        case 2: resis_value += color_index;
                                        break;


                                        case 3: if(color_index > 3)
                                                        resis_value *= pow(10, (color_index - 3));
                                                else
                                                        for(i = 0; i < (3 - color_index); i++)
                                                                resis_value /= 10;
                                }
                        }
                        else
                                no_error = 0; /* if string not found*/
                }


                if(no_error)
                        printf("Resistance value is: %.3f kilo-ohms\n\n", resis_value);
                else
                        printf("Invalid Color: %s\n\n", input_color);

                printf("Do you want to find the value of another resistor?\n => ");
                printf (" Enter y for yes. Any other character to terminate. \n");

                scanf("%c%c", &char_left, &user_choice);
                printf("\n");
          } while(toupper(user_choice) == 'y');


}




/*
function takes an input string (color) and its size and return the subscirpt of the location in
the color list. If it is not found, it returns -1
 */
int LookFor(const char COLOR_CODES [][SUB_2], const char input_color [], int size)
{


        int i, j, length,       //counters and place holder for length

        count = 0,
        found = 0,             // indicates when string is found
        location;                       // location of input_color*/


        length = strlen(input_color);

        for(i = 0; i < size && !found ; i++)
        {
                for(j = 0; j < length; j++)
                        if(COLOR_CODES[i][j] == input_color[j])
                                count++;


                if(count == length)
                        found = 1;
                else
                        count = 0;
        }

        --i;
```

```
        if(found)
                location= i;
        else
                location = NOT_FOUND;


        return location;
}
```