```
// Q1 110060007 黃俊穎
// Your program reads in two sets of time in hh:mm:ss format, where
//   hh is the hour in 24 hours per day format, i.e., 00 <= hh <= 23,
//   mm is the minute of the hour, thus, 00 <= mm <= 59,
//   ss is the second of the minute, and 00 <= ss <= 59,
// and prints out the difference, also in hh:mm:ss format.
// Example of program execution:
//
// $ ./a.out
// Time1: 22:22:22
// Time2: 11:11:11
// Difference: 11:11:11
// $ ./a.out
// Time1: 11:11:11
// Time2: 22:22:22
// Difference: 11:11:11
//
// First example has inputs: 22:22:22 and 11:11:11
// And the second example has input: 11:11:11 and 22:22:22
//
#include <stdio.h>

int main(void)
{
    int hh1, mm1, ss1;
    int hh2, mm2, ss2;
    int total_sec1;
    int total_sec2;
    int diff;
    int ans_hh, ans_mm, ans_ss;

    printf("Time1: ");
    scanf("%d:%d:%d", &hh1, &mm1, &ss1);
    printf("Time2: ");
    scanf("%d:%d:%d", &hh2, &mm2, &ss2);

    total_sec1 = hh1 * 60 ^ 2 + mm1 * 60 + ss1;
    total_sec2 = hh2 * 60 ^ 2 + mm2 * 60 + ss2;
```

```
    diff = total_sec1 - total_sec2;

    if (diff < 0) {
        diff =  diff * (-1);
    }
    ans_hh = (int)(diff / 3600);
    ans_mm = (int)((diff - ans_hh * 3600) / 60);
    ans_ss = diff - ans_hh * 3600 - ans_mm * 60;

    printf("Difference: %d:%d:%d", ans_hh, ans_mm, ans_ss);
    return 0;
}

// Q2 110060007 黃俊穎
// Given a floating point matrix, A[N][N], please write a program to find
// the sum of all fraction part of each element in A.
//
// For example, if N = 3 and
// A[N][N] = {{1.1, 2.2, 3.3},
//            {4.4, 5.5, 6.6},
//            {7.7, 8.8, 9.9}}
// then your program should execute
//
// $ ./a.out
// Sum of fraction of each element: 4.5
//
// Note that your program should be able to handle different N and A array
// and both N and matrix A are given in the source file (no need to read in).

#include <stdio.h>

#define N 3

double A[N][N] = {{1.1, 2.2, 3.3},
                  {4.4, 5.5, 6.6},
                  {7.7, 8.8, 9.9}};

int main(void)
```

```
{
    int row;                        // row variable to do first loop
    int column;                     // column variable to do second loop
    double sum = 0;                 // initialize value of sum
    double fraction_num;            // fraction of each element

    // start find each fraction of element and sum up
    for (row = 0; row < N; row++) {
        for (column = 0; column < N; column++) {
            fraction_num = A[row][column] - (int)A[row][column];
            sum += fraction_num;
        }
    }
    // print out the result
    printf("Sum of fraction of each element: %lg", sum);
    return 0;
}

// Q3 110060007 黃俊穎
// Please write a program to find all solutions for the following
// Diophantine equation with 1 <= a, b, c <= max.
//
//    a + b^3 = c^2
//
// Example program output:
//
// $ ./a.out
// Sol 1: 1 + 2^3 = 3^2
// Sol 2: 8 + 2^3 = 4^2
// ...
// Number of solutions found: xx

#include <stdio.h>

#define max 100

int main(void)
{
```

```
    int a, b, c;                // variables of the equation
    int b_max;                  // the upper bound of the variable b
    int nas = 0;                // number of solutions found

    // start finding the b's maximum in the equation
    for (b = 1; b * b * b - max * max - max < 0; b++) {
        b_max = b;              // find the maximum of b value
    }

    // start finding solutions of the equation
    for (c = 2; c <= max; c++) {
        // c starts from 2 because a + b^3 is always larger than 1
        for (b = 1; b <= b_max; b++) {
            a = c * c - b * b * b;
            if (a > 0 && a <= max) {
                // detect if a is between the valid range
                printf("Sol %d: %d + %d^3 = %d^2\n", ++nas, a, b, c);
                // show the results of all valid sets of solution
            }
        }
    }

    printf("Number of solutions found: %d\n", nas);
    // show the total number of solutions
    return 0;
}
```