Problem 1

Recall the Carbon Tax problem from last week…

The amount of Green House Gases (GHGs) emitted when a unit of fuel is burned depends on the chemical make-up of the fuel, particularly on the amount of carbon in the fuel. That fact allows for a relatively simple administrative process for applying the carbon tax.

	UNITS FOR TAX	TAX RATE JULY 1, 2012
Gasoline	¢/litre	6.67
Diesel (light fuel oil)	¢/litre	7.67
Jet Fuel	¢/litre	7.83
Natural Gas	¢/cubic metre	5.70
Propane	¢/litre	4.62

You are asked to design a program that will a read file that contains some arbitrary number of fuel usage entries, for each entry calculate the tax to be charged and at the end output the total tax to be charged to the screen.
The input file encodes each of the fuel types with the following integer values:

Fuel Type	Code
Gasoline	1
Diesel	2
Jet Fuel	3
Natural Gas	4
Propane	5

The following program needs to be refactored following the instructions within the code.

```
/*
 *  Author:   Justin Bieber
 *  Date:     January 1, 2000
 *  Purpose:  Calculate the total tax for
 *            fuel usage entries in the input file
 */

#include <stdio.h>
#include <stdlib.h>

#define _CRT_SECURE_NO_WARNINGS

#define INPUTFILE   "input.txt"
#define NUM_COLS    2

#define GAS_TAX    6.67
#define DIESEL_TAX      7.67
#define JET_FUEL_TAX    7.83
#define NATURAL_GAS_TAX 5.70
#define PROPANE_TAX     4.62

#define GAS             1
#define DIESEL          2
#define JET_FUEL        3
#define NATURAL_GAS     4
#define PROPANE         5
```

```c
int main(void) {
    FILE* inFile;

    int fuel_code;
    double amount, tax;
    double totalTax = 0;

    inFile = fopen(INPUTFILE, "r");
    if (inFile == NULL) {

        printf("Error opening input file!\n");
    } else {
        while (fscanf(inFile, "%d %lf", &fuel_code, &amount) == NUM_COLS) {
             // the following code computes the tax for a single row of the file:
             // design a function that will do perform ths calculation and
             // replace this code with a call to that function
             // Think first about what information that function must take,
             // then about what it should return
            if (fuel_code == GAS)
                tax = amount * GAS_TAX;

            else if (fuel_code == DIESEL)
                tax = amount * DIESEL_TAX;

            else if (fuel_code == JET_FUEL)
                tax = amount * JET_FUEL_TAX;

            else if (fuel_code == NATURAL_GAS)
                tax = amount * NATURAL_GAS_TAX;

            else
                tax = amount * PROPANE_TAX;

            totalTax += tax;
        }

        printf("Total tax: $%.2f\n", totalTax / 100);

        fclose(inFile);
    }

    system("PAUSE");
    return 0;
}
```

Problem 2

Recall the object detection problem from last week:
In many computer graphics programs including many games, the program must detect if two objects have come into contact with one another. For example, in a game where a user must click on an object on the screen to get points, the game must detect if the x,y coordinate of that mouse click is within the area of the object on the screen.

Your job is to write a program that will prompt a user for the coordinates of the bottom left corner of a rectangle on the screen and its width and height. The program should then read the entries from a file that contains many x,y coordinates. You can assume that each line will have one x followed by one y coordinate. You will print to the screen all of the x,y coordinates that are within the rectangle. At the end of the program print the number of total x,y coordinate pairs in the file and the number of those pairs that were within the rectangle.

The following program is incomplete; you should complete it following the instructions within the code.

Design a function called in_bounds that will take the (x,y) coordinates of a point ***and*** the (x,y) coordinates of the bottom left corner of a boundary rectangle ***and*** the width and height of the boundary rectangle.
The function should return true if the (x,y) coordinates of the point are within or on the edge of the boundary rectangle, and false otherwise.

```
/*
 *  Author:   Justin Bieber
 *  Date:     January 1, 2000
 *  Purpose:  Given a file of x,y coordinates and
 *      a bounding box coordinates and dimensions,
 *      print the x,y coordinates within the box to the screen
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>

#define _CRT_SECURE_NO_WARNINGS

#define INPUTFILE   "input.txt"
#define NUM_COLS    2



int main(void) {
    FILE* inFile;

    int x_coord_box, y_coord_box;
    int width, height;
    int x_coord, y_coord;
    int num_coords_within = 0, num_coords = 0;
```

```
    printf("Enter the x- and y-coordinates of the bottom left corner "
           "of the rectangle: \n");
    scanf("%d %d", &x_coord_box, &y_coord_box);

    printf("Enter the width and height of the rectangle:\n");
    scanf("%d %d", &width, &height);

    inFile = fopen(INPUTFILE, "r");
    if (inFile == NULL) {
        printf("Error opening input file!\n");

    } else {

        while (fscanf (inFile, "%d %d", &x_coord, &y_coord) == NUM_COLS) {
            // INCOMPLETE:
            // for each row of the file - you should call your function in_bounds
            // if the result is true, print the x_coord, y_coord to the screen
            //    and increase the count of num_coords_within


            num_coords++;
        }

        printf("Total number of pairs: %d\nNumber of pairs in rectangle: %d\n",
               num_coords, num_coords_within);

        fclose(inFile);
    }

    system("PAUSE");
    return 0;
}
```

Problem 3

Recall the object detection problem from last week:
Your aunt owns a ranch and much of the land has been flooded with torrential rain storms. These flooded areas are limited to circular and square plots of land. She has collected information for each of these plots, for the circular plots she has recorded the diameter of the affected area and for the square plots she has recorded the side length of each plot in meters. She has put all of this information into a file and sent it to you.

The input file encodes each of the land plot types with the following integer values:

Land Plot Type	Code
Circle	1
Square	2

Each entry will be of the form:

```
     Land Plot Type Code      dimension of land plot
```

The following program is incomplete; you should complete it following the instructions within the code.

```
/*
 *  Author:   Justin Bieber
 *  Date:     January 1, 2000
 *  Purpose:  Calculate the total flood area and print it to the screen
 *    print an ASCII representation of the flood area
 *    given an input file of area shape type and dimensions in meters
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define _CRT_SECURE_NO_WARNINGS

#define INPUTFILE   "input.txt"
#define OUTPUTFILE  "output.txt"
#define NUM_COLS    2

#define PI acos(-1)
#define CRCL 1
#define SQR 2

double get_area (int, double);
double square_area (double);
double circle_area (double);
void print_shape (FILE*, int, double);
void print_circle (FILE*, double);
void print_square (FILE*, double);
void print_line (FILE*, char, int);

int main(void) {
    FILE* in_file;
    FILE* out_file;

    int area_type;
    double dimension;
    double total_area = 0;
```

```
    in_file = fopen (INPUTFILE, "r");
    out_file = fopen (OUTPUTFILE, "w");

    if (in_file == NULL || out_file == NULL) {
        printf ("Error opening input or output file!\n");

    } else {

        while (fscanf (in_file, "%d %lf", &area_type, &dimension) == NUM_COLS) {

            // INCOMPLETE:
            // Call the get_area function and add the result to the total_area


            // INCOMPLETE:
            // Call the print_shape function to output the correct shape to the file


        }

        printf ("Total flooded area: %.2f m^2\n", total_area);

        fclose (in_file);
        fclose (out_file);
    }

    system ("PAUSE");
    return 0;
}

/*
 * Purpose:   gets the correct area for the given shape type
 * Parameter: the type of the shape, CRCL or SQR
 *      and the dimensions of the shape
 * Returns:   the area of the shape
 */
double get_area (int type, double dim) {

    // INCOMPLETE:









}

/*
 * Purpose:   calculate area of a circle
 * Parameter: the diameter of the circle
 * Returns:   the area of the circle
 */
double circle_area (double diameter) {
    return PI * pow(diameter, 2) / 4;
}
```

```c
/*
 * Purpose:   calculate area of a square
 * Parameter: the diameter of the square
 * Returns:   the area of the square
 */
double square_area (double sideLength) {
    return pow(sideLength, 2);
}

/*
 * Purpose:   print the correct shape type to a file
 * Parameter: the file to print to, the type of shape,
 *      and the dimesions of the shape
 * Returns:   void
 */
void print_shape (FILE* f_out, int type, double dim) {

    // INCOMPLETE:









}

/*
 * Purpose:   prints to the file a line of specified character,
 *      the specified times with no newline at the end
 * Parameter: the character to print, the number of times to print is
 * Returns:   void
 */
void print_line (FILE* f_out, char c, int num){
    int count = 0;

    while (count < num) {
        fprintf (f_out,"%c", c);
        count++;
    }
}

/*
 * Purpose:   prints a square of specified length to a file
 *      with an extra newline after the square
 * Parameter: the file to print to, the diameter of the circle
 * Returns:   void
 */
void print_square (FILE* f_out, double length) {
    int limit = length;
    int row = 0;

    while (row < limit) {
        print_line(f_out, 'x', limit);
        fprintf (f_out, "\n");
        row++;
    }

    fprintf (f_out, "\n");
}
```

```c
/*
 * Purpose:   prints to a file a circle of specified
 *      diameter rounded up to an odd number,
 *      with an extra newline after the square.
 * Parameter: the file to print to, the diameter of the circle
 * Returns:   void
 */
void print_circle (FILE* f_out, double diameter) {
    int row, num_spcs, num_strs;
    int limit = diameter;
    int top_half = limit/2 + 1;
    int bot_half = limit/2;

    if (limit%2 == 0)
        limit++;

    row = 0;
    while (row < top_half) {
        num_spcs = limit/2 - row;
        print_line (f_out, ' ', num_spcs);

        num_strs = row * 2 + 1;
        print_line (f_out, '*', num_strs);

        fprintf (f_out, "\n");

        row++;
    }

    row = 1;
    while (row <= bot_half) {
        num_spcs = row;
        print_line(f_out, ' ', num_spcs);

        num_strs = limit - row * 2;
        print_line (f_out, '*', num_strs);

        fprintf (f_out, "\n");

        row++;
    }

    fprintf (f_out, "\n");
}
```