1806ICT Programming Fundamentals

Week 9 : Structures

1. Given a structure that stores the x and y coordinates for a point on the X-Y Cartesian plane:

```
struct point {
      double x;
      double y;
};
typedef struct point Point;
```

   Use the struct in your program to
   a. Write a function to compute the Euclidean distance between two points
   b. Write a function that returns 1 if the two points are "equal", and 0 otherwise. Since the x and y coordinates are floating point values, use a tolerance value of 0.000001 in your comparison
   c. Write a main program to test your functions

2. Define a `Rect` type for rectangles that are parallel to the x-axis and y-axis in the X-Y Cartesian plane. A rectangle is represented by its lower left and upper right endpoints, with the points defined using the `Point` type. Using `Rect` in your program,
   a. Write a function to compute the area of a rectangle
   b. Write a function that returns 1 if a point falls within a rectangle, and 0 otherwise. This function will take in two parameters, a `Point` variable and a `Rect` variable.

   Write a main program to test your functions

3. Write a program that computes the area and circumference (or perimeter) for a variety of geometric figures. You can use the following definitions of structure types for a circle, square, and rectangle, and a definition of a union type with a component of each figure type.

```
typedef struct
{     Point top_left;
      double area;
      double perimeter;
      double side;
} squareType;

typedef struct
{     Point top_left;
      Point bottom_right;
      double area;
      double perimeter;
} rectangleType;

typedef struct
{
      Point centre;
      double area;
```

```
        double circumference;
        double radius;
       squareType bounding_box;
} circleType;


typedef union
{
        circleType circle;
        squareType square;
        rectangleType rectangle;
} figureData;

typedef struct
{
   char shape;   // denotes the correct interpretation of the union
   figureData fig;
} figureType;
```

The `char` variable `shape` can be used to identify the geometric figure for which the computation of area and circumference (or perimeter) is being done.

Your program will ask the user to enter either `c` (for circle), `s` (for square) and `r` (for rectangle) and the corresponding dimensions for those geometric figures. It should also have at least the following functions:

```
figureType computeArea(figureType object)
figureType computePerimeter(figureType object)
void printFigure(figureType object)
bool intersect(figureType object1, figureType object2)
```

Sample run:

Input	Output
c 1 2 2	Area of circle = 12.57, Perimeter of circle = 12.57
s 3 4 3	Area of square = 9, Perimeter of square = 12
r 5 6 1 5	Area of rectangle = 5, Perimeter of rectangle = 12
s 3 4 3 r 5 6 1 5	Yes, they overlap

4. Given a structure that stores a date as follows:

```
struct date {
      int day;
      int month;
      int year;
};
typedef struct date Date;
```

Write a program that reads in a date, and prints out the next day's date. You should write the following two functions in your program to help you do the computation.

The first function `isLeapYear` determines whether the year is a leap year. This function takes in one parameter, a `Date` variable. A year is a leap year if

- It is a multiple of 4 but not a multiple of 100

OR

- It is a multiple of 400

For example, 1996 and 2000 are leap years, but 1900, 2002, and 2100 are not.

The second function `numberOfDays` determines how many days there are in a given month for a given year. This function takes in one parameter, a `Date` variable. This function should also call the `isLeapYear` function to help it determine how many days there are in February.

Sample run:

Input	Output
31 3 2017	1/4/2017
28 2 2016	29/2/2016
28 2 2015	1/3/2015

5. Given a structure that stores a time (in 24-hour format) as follows:

```
struct time {
      int hour;
      int minutes;
      int seconds;
};
typedef struct time Time;
```

Write a function `elapsed_time` that takes as its arguments two `Time` structures and returns a `Time` structure that represents the elapsed time in (hours, minutes, and seconds) between the two times. Write a main program to test your function.

Sample run:

Input	Output
3 45 15 9 44 3	5 hours 58 minutes 48 seconds