ECE 71/191T – Data Structures and Algorithms

Dr. Gregory R. Kriehn, Fresno State
C++ Homework Assignment: Chapter 10

Assignment Due By: Class on Thu, Feb 16

HOMEWORK #18 – Header Files

Split the user-defined functions from HW #17 into two user-defined **`.cpp`** source files and **`.h`** header files, respectively. Place the functions that update information about the team or player into an **`updatePlayer.cpp`** file and all other functions (aside from the main function) into an **`updateData.cpp`** file. Create the appropriate header files, as necessary.

Re-compile your code and make adjustments until the program works identically to HW #17.

HOMEWORK #19 – Coordinate Transformations

Create a menu-driven program that prompts a user to enter a 3D (x, y, z) point into a Cartesian coordinate system and allows the user to perform various transformations on the data point.

Specifications:
Specify, design, and implement a class called **`cartesian`** that can be used to keep track of the position of a point in 3D space. Include member functions to clear the (x, y, z) data point, set the point, get the data point from the class object, and translate the point a given amount along any of the three axes. Also provide member functions that rotate the point by an angle around a specified axis.

To compute the rotations, use trigonometry. Suppose you have a point with coordinates x, y, and z. After rotating the point counter-clockwise by an angle θ, the point will have new coordinates x', y', and z'. The equations for the new coordinates use the **`cmath`** library functions **`sin`** and **`cos`**:

After a θ rotation around the x-axis:

$x' = x$
$y' = y \cos(\theta) - z \sin(\theta)$
$z' = y \sin(\theta) + z \cos(\theta)$

After a θ rotation round the y-axis:

$x' = x \cos(\theta) + z \sin(\theta)$
$y' = y$
$z' = -x \sin(\theta) + z \cos(\theta)$

After a θ rotation around the *z*-axis:

$$x' = x \cos(\theta) – y \sin(\theta)$$
$$y' = x \sin(\theta) + y \cos(\theta)$$
$$z' = z$$

As an example, if the current (*x*, *y*, *z*) data point is (0,1,0), a rotation around the x-axis by 90° will move the data point to (0,0,1).

Create a class file called **`cartesian.cpp`** and an appropriate **`cartesian.h`** header file. Use the following public class member functions:

```
    void clear();
    // Clears the x,y,z data point

    void set(double numX, double numY, double numZ);
    // Sets the x,y,z data point to a specific set of values

    void get(double &numX, double &numY, double &numZ) const;
    // Gets the current x,y,z data point

    void translate(double dx, double dy, double dz);
    // Translates the x,y,z data point by dx, dy, and dz along
    // their respective axes

    void rotateX(double thetaX);
    // Rotates the data point around the x axis by theta

    void rotateY(double thetaY);
    // Rotates the data point around the y axis by theta

    void rotateZ(double thetaZ);
    // Rotates the data point around the z axis by theta

    caresian();
    // Default constructor

    Cartesian(double numX, double numY, double numZ);
    // Constructor with initial values
```

Finally, use the following private members:

```
    double x;
    double y;
    double z;
```

The program should also include the following user-defined functions in the **`main.cpp`** file:

```
void printCoordinate(const double x, const double y, const
     double z);
// Prints the x,y,z coordinate to the screen

void printMenu();
// Prints the menu to the screen

void adjustCoordinate(cartesian &coordinate);
// Prompts the user for input and performs the
// corresponding transformation on the data point
```

Use the following menu options:

```
Coordinate Menu
  (1)  Clear Coordinate
  (2)  Set Coordinate
  (3)  Translate Coordinate
  (4)  Rotate Coordinate
  (5)  Print Coordinate
  (6)  End Program

Input Selection: 5

Current Cartesian Coordinate:
  x = 0.00
  y = 0.00
  z = 0.00
```

The menu should always be re-displayed after printing the current coordinate to the screen. Use the following prompts for options (2) – (4), respectively:

```
Enter (x, y, z) coordinate:
Enter (dx, dy, dz) translation vector:
Enter (theta_x, theta_y, theta_z) rotation vector:
```

Ensure that your program is well documented using the format specified by Malik, including the use of Unified Modeling Language when describing your class members for your pseudocode. E-mail your code to myself and grader once your program works.

HOMEWORK #20 – Tic-Tac-Toe Game

Write a program that allows two players to play the Tic-Tac-Toe game.

Specifications:
Your program must contain the class **`TicTacToe`** to implement a TicTacToe object. Include a 3x3 2D array as a private member variable, as well as a variable to store the number of moves as private member variables. Include additional public members:

```
void play();
// Function to play the game

void displayBoard() const;
// Function to print the board

bool isValidMove(int row, int col) const;
// Function to determine if a move is valid

bool getMove(char moveSymbol);
// Function to get a move for a player

status gameStatus();
// Function to determine the current status of the game
// Uses the status enumeration data type for members WIN,
// DRAW, and CONTINUE

void restart();
// Function to restart the game

TicTacToe();
// Default constructor
```

Use **`system(“clear”)`** to call the operating system’s clear screen command, as necessary.

```
     Tic-Tac-Tow Game

         1   2   3
           |   |
      1    |   |
        ___|___|___
           |   |
      2    |   |
        ___|___|___
           |   |
      3    |   |
           |   |

Player X Enter Move: 1 2
```

```
     Tic-Tac-Tow Game

         1   2   3
           |   |
      1    | X |
        ___|___|___
           |   |
      2    |   |
        ___|___|___
           |   |
      3    |   |
           |   |

Player O Enter Move: 1 2
Invalid Move

Player O Enter Move: 2 1

     Tic-Tac-Tow Game

         1   2   3
           |   |
      1    | X |
        ___|___|___
           |   |
      2  O |   |
        ___|___|___
           |   |
      3    |   |
           |   |
```

After the game is played print one of the following:

```
     Player [X/O] wins!
     The game is draw!
```

Prompt the user to play the game again, as necessary.

Ensure that your program is well documented using the format specified by Malik, including the use of Unified Modeling Language when describing your class members for your pseudocode. E-mail your code to myself and grader once your program works.