ECE 71 – Engineering Computations in C

Professor Kriehn – Fall 2017

Code Due By: Midnight on Friday, Dec 1
Writeup Due By: Class on Mon/Tue, Dec 4/5

HOMEWORK #40 – Structures and Football

It's nearing playoff season in football – time to keep track of some statistics for everyone's favorite team, the Packers!

Specifications:
Write a program that declares a **`struct`** called **`footBallPlayerType`** to store the data of a football player (player's last name, player's position, number of touchdowns, number of catches, number of passing yards, number of receiving yards, and the number of rushing yards). Declare an array of structures to store the data of 10 football players.

Your program must contain functions to input data from a file and output data to a file. The input and output files are called "**`PackersInput.txt`**" and "**`PackersOutput.txt`**", respectively. A copy of the input file can be found on the website. Add functions to print menu options, print all of the team data, print a single player's data, search the array to find the index of a specific player, update the data of a player based on the number of touchdowns, catches, etc. Before the program terminates, give the user the option to save the data. Assuming **`N`** is a constant, use the following function prototypes:

```
// Display the menu
void showMenu();

// Get team data from the input file
void getData(ifstream &inputFile, footBallPlayerType list[N]);

// Save current team data to output file
void saveData(ofstream &outputFile, footBallPlayerType list[N]);

// Print entire team data to the screen
void printData(footBallPlayerType list[N]);

// Print a single player's data to the screen
void printPlayerData(footBallPlayerType list[N], int playerNum);

// Search the data for a player
int searchData(footBallPlayerType list[N], string n);


// Update the number of touchdowns for a player
void updateTouchDowns(footBallPlayerType list[N],
       int tDowns, int playerNum);

// Update the number of catches for a player
```

```
void updateCatches(footBallPlayerType list[N], int catches,
  int playerNum);

// Update the number of passing yards for a player
void updatePassingYards(footBallPlayerType list[N],
  int passYards, int playerNum);

// Update the number of receiving yards for a player
void updateReceivingYards(footballPlayerType list[N],
  int recYards, int playerNum);

// Update the number of rushing yards for a player
void updateRushingYards(footBallPlayerType list[N],
  int rushYards, int playerNum);
```

If you execute the program, the following information should be displayed:

```
~> main.o
Select one of the following options.
  (1)  Print the Team's Data
  (2)  Print a Player's Data
  (3)  Update a Player's Touchdowns
  (4)  Update a Player's Catches
  (5)  Update a Player's Passing Yards
  (6)  Update a Player's Receiving Yards
  (7)  Update a Player's Rushing Yards
 (99)  Quit the Program

Input Selection: 1

Name    Position Touchdowns Catches Passing Receiving Rushing

Rodgers       QB         13       0    1385         0      83
Hundley       QB          5       0    1185         0     110
Jones         RB          3       8       0        16     370
...


Select one of the following options.
   (1)  Print the Team's Data
   (2)  ...

Input Selection: 2

Enter player's name: Rodgers

Name: Rodgers
  Position: QB
  Touchdowns: 13
  Number of Catches: 0
  Passing Yards: 1385
  Receiving Yards: 0
  Rushing Yards: 83


Select one of the following options.
   (1)  Print the Team's Data
   (2)  ...

Input Selection: 3

Enter player's name: Rodgers

Enter number of touchdowns added: 7


Select one of the following options.
   (1)  Print the Team's Data
   (2)  ...

Input Selection: 2

Enter player's name: Rodgers

Name: Rodgers
  Position: QB
  Touchdowns: 20
  ...


Select one of the following options.
   (1)  Print the Team's Data
   (2)  ...

Input Selection: 4

Enter player name: Billy

Enter number of catches to be added: 3

Invalid player number.


Select one of the following options.
   (1)  Print the Team's Data
   (2)  ...

Input Selection: 99
```

```
Would you like to save data (Y/N): Y

~> more PackersOutput.txt
Bill QB 77 0 8754 0 57
Jackson RC 55 87 50 5490 574
...
```

Once you verify the operation of your program, submit your source code to the Grader Program.

HOMEWORK #41 – 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 two private data member variables – a 3x3 2D character array to store information about the board, and a variable to store the number of moves. 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
```

Notice the `gameStatus()` function returns a "`status`" data type, since there are 3 outcomes for the game after every round: (1) Either player X or O wins, (2) the game is a draw, (3) or after a move no one wins, the game is not a draw, and needs to continue. Therefore, you should create a global enumerated data type called "`status`" with values `WIN`, `DRAW`, and `CONTINUE` as use them appropriately in your function.

Finally, use **`system("clear")`** to call the operating system’s clear screen command to clear the screen whenever necessary.

If you execute the program, the following information should be displayed:

~> **main.o**

```
     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!
```

In other words if Player X wins, print:

```
     Player X wins!
```

If player O wins, print:

```
     Player O wins!
```

If the game is a draw, print:

```
     The game is a draw!
```

Then prompt the user to play the game repeatedly after a win or draw, as necessary. Use:

```
     Play Again (Y/N)?:
```

Once you verify the operation of your program, submit your source code to the Grader Program.