COMP500 / ENSE501: Week 7 – Exercise:

EXERCISE NAME: *Modular R-P-S*

This exercise builds a simple "Rock-Paper-Scissors" game using modularity.

Given the following source code:

```
// Student ID:
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>

// TODO: Part a) Declare enumerations here:

// Function Prototypes:
enum Choice convert_input_to_choice(char player_input);
enum Choice get_ai_choice(void);
enum Choice get_player_choice(void);
enum Boolean is_draw(enum Choice ai, enum Choice player);
enum Boolean is_player_win(enum Choice ai, enum Choice player);
enum Boolean play_again(void);
void print_choice(enum Choice choice);
void print_game_over(int ai_wins, int player_wins, int draws);
void print_player_turn_prompt(void);
void print_players_move(enum Choice player);
void print_ais_move(enum Choice ai);
void print_result(enum Result result);
void print_stats(int ai_wins, int player_wins, int draws);
void print_welcome(void);

// The main function definition:
int main(void)
{
    srand(time(0));

    // TODO: Part c) Insert main code here:

    return 0;
}

// TODO: Part b) Define functions here:
```

Part a:

Write your Student ID in the C comment at the top of the source code file.

Below the `TODO: Part a)` comment, declare three different enumerated types:

1. **`Boolean`**: has the values: **`FALSE`**, **`TRUE`**
2. **`Result`**: has the values: **`DRAW`**, **`AI_WINS`**, **`PLAYER_WINS`**
3. **`Choice`**: has the values: **`INVALID_CHOICE = -1`**, **`ROCK`**, **`PAPER`**, **`SCISSORS`**, **`MAX_CHOICE`**

Part b:

Below the `TODO: Part b)` comment, implement the following function definitions.

Firstly, implement the function definition of **`print_welcome`** such that when called it would output the following to the console:

```
Rock-Paper-Scissors!
====================
```

Next, implement the function definition of **`print_player_turn_prompt`** such that when called it would output the following to the console:

```
Your choice... rock(r), paper(p), scissors(s)?
```

Next, implement the function definition of **`print_stats`** such that when called it would output the following to the console, where the **`?`** symbols are replaced with integer values based upon the three parameters passed into the function:

```
AI Wins: ?, Player Wins: ?, Draws: ?
```

Next, implement the function definition of **`print_game_over`** such that when called it would output the following to the console:

```
The final stats are...
```

After outputting the text above, **`print_game_over`** must then call **`print_stats`** passing in its parameters as the arguments for the **`print_stats`** call.

Next, implement the function definition of **`print_choice`**, which must print the following text based upon the parameter passed into the function:

Input **`enum`** Parameter	**`printf`** Text Output
`ROCK`	**`Rock`**
`PAPER`	**`Paper`**
`SCISSORS`	**`Scissors`**

An example of console output when the **`print_choice`** function is called with **`ROCK`** as an argument is as follows:

```
Rock
```

Next, implement the function definition of **convert_input_to_choice**, which must return an **enum Choice** value, based upon the **char** parameter passed into the function as follows:

Input **char** Parameter	**enum Choice** returned
'r'	ROCK
'R'	ROCK
'p'	PAPER
'P'	PAPER
's'	SCISSORS
'S'	SCISSORS

Next, implement the function definition of **get_ai_choice**, which must return a random **enum Choice** value, use the following pseudo code to implement the function:

```
START get_ai_choice
    DECLARE enum Choice ai
    ai = rand() % MAX_CHOICES
    return ai
END
```

Next, implement the function definition of **is_draw**, which must return an **enum Boolean** value, use the following pseudo code to implement the function:

```
START is_draw(enum Choice ai, enum Choice player)
    DECLARE enum Boolean result AS FALSE
    IF ai == player
        SET result TO TRUE
    ENDIF
    RETURN result
END
```

Next, implement the function definition of **is_player_win**, which must which must return an **enum Boolean** value, use the following pseudo code to implement the function:

```
START is_player_win(enum Choice ai, enum Choice player)
    DECLARE enum Boolean player_wins AS FALSE
    IF ai MATCHES ROCK AND player MATCHES PAPER
        SET player_wins TO TRUE
    IF ai MATCHES PAPER AND player MATCHES SCISSORS
        SET player_wins TO TRUE
    IF ai MATCHES SCISSORS AND player MATCHES ROCK
        SET player_wins TO TRUE
    ENDIF

    RETURN player_wins
END
```

Next, implement the function definition of **print_players_move**, based upon the following pseudo code:

```
START print_players_move(enum Choice player)
    PRINT "The player chose: "
    CALL print_choice WITH ARGUMENT player
    PRINT newline
END
```

Next, implement the function definition of **print_ais_move**, based upon the following pseudo code:

```
START print_ais_move(enum Choice ai)
    PRINT "The AI chose: "
    CALL print_choice WITH ARGUMENT ai
    PRINT newline
END
```

Next, implement the function definition of **print_result**, based upon the following pseudo code:

```
START print_result(enum Result result)
    IF result MATCHES DRAW
        PRINT "DRAW!\n"
    ELSEIF result MATCHES PLAYER_WINS
        PRINT "Player wins!\n"
    ELSEIF result MATCHES AI_WINS
        PRINT "AI wins!\n"
    ENDIF
    PRINT newline
END
```

Next, implement the function definition of **play_again**, which returns an **enum BOOLEAN** value, based upon the following pseudo code:

```
START play_again
    DECLARE enum Boolean another_round
    SET another_round TO FALSE
    DECLARE char input
    SET input TO 0

    PRINT "Play again (y/n)? "
    READ input
    IF input MATCHES 'y'
        SET another_round TO TRUE
    ENDIF

    RETURN another_round
END
```

Next, implement the function definition of **get_player_choice**, which returns an **enum CHOICE** value, based upon the following pseudo code:

```
START get_player_choice
    DECLARE char input
    SET input TO 0
    READ input;
    DECLARE enum Choice choice
    SET choice TO convert_input_to_choice(input)

    RETURN choice
END
```

Part c:

In the **main** function, under the comment **TODO: Part c)** convert the following pseudo code into source code:

```
BOOLEAN playing = TRUE
SET ai_win_count AS 0
SET player_win_count AS 0
SET draw_count AS 0
CALL print_welcome
WHILE (playing == TRUE)
    CALL print_stats(ai_win_count, player_win_count, draw_count)
    Choice ai_choice = get_ai_choice()
    CALL print_player_turn_prompt
    Choice player_choice = get_player_choice()
    PRINT newline
    print_players_move(player_choice)
    PRINT newline
    print_ais_move(ai_choice)
    PRINT newline
    IF (is_draw(ai_choice, player_choice) == TRUE)
        print_result(DRAW)
        INCREMENT draw_count
    ELSE IF (is_player_win(ai_choice, player_choice) == TRUE)
        print_result(PLAYER_WINS)
        INCREMENT player_win_count
    ELSE
        CALL print_result(AI_WINS)
        INCREMENT ai_win_count
    ENDIF
    playing = play_again()
    PRINT newline
ENDWHILE
CALL print_game_over(ai_win_count, player_win_count, draw_count)
```

Compile and test your implementation, ensure your program functions as expected.

An example play through of the game is as follows:

```
Rock-Paper-Scissors!
====================

AI Wins: 0, Player Wins: 0, Draws: 0

Your choice... rock(r), paper(p), scissors(s)? p

The player chose: Paper

The AI chose: Scissors

AI wins!

Play again (y/n)? y

AI Wins: 1, Player Wins: 0, Draws: 0

Your choice... rock(r), paper(p), scissors(s)? p

The player chose: Paper

The AI chose: Scissors

AI wins!

Play again (y/n)? y

AI Wins: 2, Player Wins: 0, Draws: 0

Your choice... rock(r), paper(p), scissors(s)? r

The player chose: Rock

The AI chose: Scissors

Player wins!

Play again (y/n)? n

The final stats are... AI Wins: 2, Player Wins: 1, Draws: 0
```