The game of **Rock Paper Scissors** is played by two players. Each player chooses one of the three options: Rock, Paper or Scissors. The players reveal their choices at the same time. The winner is the player whose choice dominates the other player’s choice. Rock will dominate Scissors. Scissors will dominate Paper. Paper will dominate Rock. Otherwise, the match is a tie.

### Assignment

Create a program that allows the user to play Rock, Paper, Scissors against a random number generator. Your program must allow the user to keep playing as long as they choose to do so. Each match, it will randomly choose one of the three options for the random player, ask the user for their choice and determine whether the user or the random player wins.

### Requirements

Your program:

- Must use pseudo-random numbers to choose the computer option.
- Must correctly determine the winner of each match, or tie.
- Must track the number of wins, losses, and ties, displaying the current statistics each round of play and at the end.
- Must prompt the user whether to continue or quit before each match.
- Must force the user to select one of the three legal options. The user must type exactly `rock`, `scissors` or `[paper]`. That is all lower-case.
- Must not crash if the user gives a non-legal input.
- Must be called `main`.

### Instructions

Create your rock, paper, scissors functions in `rps.cpp` with function declarations in `rps.h`. Create the main program in `main.cpp`.

Your assignment requires the creation of the following functions. The functionality for each function is described below. You must follow the specifications exactly, but may choose your own method for solving the problem described for each. Once you have completed a function you should run the unittest for that function and have it pass all tests. Fix any errors, warnings, and/or failures.

- **`void initializeRandomNumbers()`**: This function must initialize the pseudo-random number system using the current time. Eventually, your main function will call this function at the beginning of the program.
- **`int getRandomPlayerChoice()`**: This function must return a randomly selected integer. `1 == ROCK, 2 == PAPER, 3 == SCISSORS`
- **`int getUserPlayerChoice(std::istream& input_stream, std::ostream& output_stream)`**: This function must prompt the user for a choice between rock, paper and scissors. The user must type one of those strings. The function will then return an integer that corresponds to the user’s choice, using the same system as the previous function. If the user gives any response other than the 3 acceptable options, the user must be prompted again, until a valid choice is selected. The function must read from `input_stream` and write to `output_stream` for communication with the user.
- **`int determineWinner(int user_choice, int random_choice)`**: This function will return an integer `11 == USER, 12 == RANDOM_PLAYER, 13 == TIE` for which player won the match. `user_choice` and `random_choice` are values as described in `getUserPlayerChoice()`. If either `user_choice` or `random_choice` is not valid, return `0`.
- **`void displayMatchResults(std::ostream& output_stream, int user_choice, int random_choice, int winner)`**: This function must display a message to `output_stream` that contains information on what the user’s choice was, what the random player’s choice was and who won. A sample of the format is `You chose rock. Computer chose scissors. You win.`.
- **`void displayStatistics(std::ostream& output_stream, int number_user_wins, int number_user_losses, int number_user_ties)`**: This function must display a message to `output_stream` that contains information on how many matches the user has won, lost and tied. Sample format is `Win: 3 Lose: 2 Tie: 1`.
- **`bool askUserIfGameShouldContinue(std::istream& input_stream, std::ostream& output_stream)`**: This function must display a prompt to the `output_stream` asking if the user would like to continue. This function must read from the `input_stream`. If the word read begins with a `y` or a `Y`, then the function must return `true`. Any other input should cause the function to return `false`. 

---

### Example Code Snippets

```cpp
// Example of `initializeRandomNumbers`
void initializeRandomNumbers()
{
    // Initialize pseudo-random numbers
}

// Example of `getUserPlayerChoice`
int getUserPlayerChoice(std::istream& input_stream, std::ostream& output_stream)
{
    // Prompt user for choice
    // Read input from `input_stream`
    // Write output to `output_stream`
    // Return choice integer
}

// Example of `determineWinner`
int determineWinner(int user_choice, int random_choice)
{
    // Determine winner
    // Return winner integer
}

// Example of `displayMatchResults`
void displayMatchResults(std::ostream& output_stream, int user_choice, int random_choice, int winner)
{
    // Display match results
    // Write output to `output_stream`
}

// Example of `displayStatistics`
void displayStatistics(std::ostream& output_stream, int number_user_wins, int number_user_losses, int number_user_ties)
{
    // Display statistics
    // Write output to `output_stream`
}

// Example of `askUserIfGameShouldContinue`
bool askUserIfGameShouldContinue(std::istream& input_stream, std::ostream& output_stream)
{
    // Ask user if game should continue
    // Read input from `input_stream`
    // Write output to `output_stream`
    // Return true if user wants to continue
    return false;
}
```
**Example Execution**

```
./main
Choice? (rock, paper, scissors) laser
That is not a valid option.
Choice? (rock, paper, scissors) rock
Win: 0 Lose: 1 Tie: 0
Play again? Yup!
Choice? (rock, paper, scissors) rock
Win: 1 Lose: 1 Tie: 0
Play again? yes
Choice? (rock, paper, scissors) rock
Win: 1 Lose: 1 Tie: 0
Play again? no
```

**Show Off Your Work**

The unit tests for this assignment will be available in CodeGrinder. They only allow you to test the functions listed above, they do not have tests for the main program.

To receive credit for this assignment, you must upload the source code (.cpp and .h files) and the Makefile to the Canvas submission system.

Additionally, the program must build, run and give correct output.