Rock-Paper-Scissors-Lizard-Spock

Description

This project will test the understanding of the material that you have learned thus far. You will **design** and implement a game program called, "Rock, Paper, Scissors, Lizard, Spock". This program will serve as your first lab exam but will be referred to as a project.

Game Rules

The game is an expansion on the game Rock, Paper, Scissors. Each player picks an object and reveals it at the same time using hand gestures to replicate a rock, paper, or scissors. The winner is the one who defeats the other. In a draw (tie), the game is declared a draw and another game is played.

- Scissors cuts paper.
- Paper covers rock.
- Rock crushes lizard.
- Lizard poisons Spock.
- Spock smashes scissors.
- Scissors decapitates lizard.
- Lizard eats paper.
- Paper disproves Spock.
- Spock vaporizes rock.
- Rock crushes scissors.

Algorithm

Game start:

1. Display the name of the game and the rules as listed above.
2. Ask the user if they would like to play the game.
   i. If no, then exit.
3. The computer chooses a number at random between 1 and 5 which corresponds to Rock, Paper, Lizard, Scissors, Spock, respectively.
4. The computer displays a menu of those choices to the user as a prompt for the user's choice.
5. Print the computer's choice and the user's choice.
6. The computer compares its number with the user's and declares a winner or a draw.
7. Go to 2

Sample Output (user input is in bold)

```
Would you like to play Rock-Paper-Scissors-Lizard-Spock? (y/n): Y
  (R)ock
  (P)aper
  (S)cissors
  (L)izard
  Spoc(k)
Enter the letter of your choice: K
You chose Spock the computer chose Lizard
Lizard Poisons Spock
You lose!

Play again? (y/n): Y
```

```
   (R)ock
   (P)aper
   (S)cissors
   (L)izard
   Spoc(k)
Enter the letter of your choice:
```

Requirements

- Your program **must be highly modularized**. Your main function should consist of a asking the user if they want to play then calling a module to play the game if they answer yes and exiting if they do not.
- Write a function named, "turn", that contains a switch statement that calls a function based on the computer's random choice. This function should accept as input, the user's choice (in the form of a character).
  - E.g., if the computer chose Rock, call a function called "rock" and pass in as an argument the player's choice.
  - The function called, "rock", should determine who the winner was, print the proper rule and print if the user won, lost, or it was a draw. An example is: if the computer chose "rock" and the user chose "Spock" then the function *rock* is called with *k* as an argument and should print "Spock vaporizes rock", then print, "You win!".
- Write five functions (called *rock, paper, scissors, lizard,* and *spock*) as described above that take as input the user's choice (a character). It then prints the rule that applies, and whether the user won, lost, or there was a draw.
- Write a function called getRandomChoice that takes no input (has no parameters) and returns a random number between 1 and 5 inclusive. This number represents the computer's choice of rock, paper, scissor, lizard, Spock respectively.
- **Be sure to validate user-input.** If the user enters an invalid choice, then you should inform the user that it is invalid and prompt again.
- You must implement and use the control structures and functions listed above but you can also implement any others that you might find useful (i.e., you might want to write a function that validates user input (takes the input as an argument and returns true or false if the input is valid or not).
- All user input will be in the form of a character. The user should be allowed to enter the uppercase and lowercase form of the character.
- You must perform input validation. If the user enters an option other than those allowed, display "bad option" and prompt again. This must be done for any prompt.
- At the end of a turn, you will ask the user if they want to play again. If they answer yes, then restart the game by printing the menu of options, if no, exit the game. **The initial greeting should not print again.**

Activity

1. Implement a C++ program by following the above requirements strictly.
   - Call the program "rpsls_game.cpp"
   - Use the lab template file for this program and update the comments at the top.
     Hint: there is nothing wrong with implementing and testing functions before you write the menu system (user interface). This is a common practice amongst programmers.
2. Write a report on your experience with this project. The report should be named, "<your mtsac username>_project-1_report"; where "<your mtsac username>" is replaced with your Mt. SAC username.
   a. Include a cover page with the following lines:
      - project number and name

- your name
- your Mt SAC email address,
- your instructor's name (Prof. D. Atanasio)

The cover page should use a 12-point Times New Roman font with this information centered with a line spacing of 2.

b. The rest of the report should contain an "introduction" to the project where you explain the game and its rules. Write this in your own words.

c. You should then write a section called "design and Implementation" where you discuss your design decisions and the problems that you encountered while creating your program, and the steps that you took to solve those problems. Do not include actual code in this section or anywhere in the report.

d. Your report should be at least two pages plus the cover page. It should have 1-inch margins and be in Times New Roman 12-point font with 1.5 line spacing.

What to Submit

- Your report as a PDF (<your MtSAC username>_project-1_report.PDF)
- Your source code file (rpsls_game.cpp)

Hint: the sooner you get started the more you will enjoy this project and the better you will do.

Reminder

You are responsible to do your own work. **This is not a team project**. **Do not show anyone your code and do not look at, or copy, any code from any source (even the lecture notes or the book); create your own code.** Our lectures and the book contain all the information you need to complete this project. Any violation of the school's academic integrity policy or the policy of this class will result in a zero grade and an academic misconduct report filed with the school; no excuse will be accepted.

Your program must compile and run to receive any credit. If I cannot compile your program, then you will receive a zero for your score. Treat this like any other exam, start right away and put effort into it.

Rubric

This project/exam is worth 200 points. The points will be distributed according to the chart below. If the program does not compile or is missing, then no points will be awarded.

Report		**50**
Formatting	**20**	
Content	**30**	
Program		**150**
Requirements	**40**	
Correctness	**70**	
Code Quality	**40**	
Total		**200**