Group Project

Submit Assignment

Due Nov 5 by 11:59pm **Points** 50 **Submitting** a file upload
File Types zip

Play Rock, Paper, Scissors!!!

For this task, you need to work as a group to finish this project. First, form the project group. Then discuss frequently inside the group using whatever software you like and divide the work. You must finish the program and the report on time and submit them as a zip file on Canvas. It is challenging to work together online. Be prepared and start early.

Sign up in one group here.

You can only join one group. Each group allows 2 to 5 studetns. You must sign in one group in order to get the grade. Please also includes all the group member names in the final report (a pdf file). Name your zip file by the group number (Group1.zip, etc.). Each group only need to submit **one copy** to Canvas. So please communicate well to avoid multiple submissions. This project is due at the end of week 6 and doesn't allow late submission.

Requirement (Idea from the MIT OpenCourseWare):

Tool Class:

You will implement a class called **Tool**. It should have an int field called strength and a char field called type. **You may make them either private or protected.** The Tool class should also contain the function void SetStrength(int), which sets the strength for the Tool.

Rock, Paper, and Scissors Class:

Create 3 more classes called **Rock**, **Paper**, and **Scissors**, which **inherit from Tool**. Each of these classes will need a default constructor that sets the strength to 1 and a non-default constructor which will take in an int used to initialize the strength field. The constructor should also initialize the type field using 'r' for Rock, 'p' for Paper, and 's' for Scissors. These classes will also need a public function fight(Tool) that compares their strengths in the following way:

Rock's strength is doubled (temporarily) when fighting scissors, but halved (temporarily) when fighting paper.

In the same way, paper has the advantage against rock, and scissors against paper.

The strength field shouldn't change in the function.

You may also include any extra auxiliary functions and/or fields in any of these classes.

RPSGame Class:

In addition, you will create a class called **RPSGame**, which allows a **human to play the rock, paper, scissors game against the computer.** Your RPSGame must have two Tool* for the human and computer tool because you don't know the new tool they'll select with each round. The RPSGame should also have three int fields to keep track of the number of human_wins, computer_wins, and ties.

You can choose the strategy for the computer guesses, but it cannot be based on what the human selected for a tool in the current game!!! Example of a novice and veteran computer AI:
http://www.nytimes.com/interactive/science/rock-paper-scissors.html?_r=0 (http://www.nytimes.com/interactive/science/rock-paper-scissors.html?_r=0)

After the human selects the tool for the current game, display the computer's tool, a message describing who won, the current stats for the wins and ties, and then ask the user if he/she wants to play again.

Example Play of Game:

Welcome to Rock, Paper, Scissors! Do you want to choose different strengths for the tools? (y-yes, n-no) n

Choose your tool (r-rock, p-paper, s-scissor, e-exit): r

Computer chose scissor.

You win!!!

Human wins: 1

Computer wins: 0

Ties: 0

Choose your tool (r-rock, p-paper, s-scissor, e-exit): r

Computer chose paper.

Computer wins! :-(

Human wins: 1

Computer wins: 1

Ties: 0

Choose your tool (r-rock, p-paper, s-scissor, e-exit): e

Things to consider:

If you choose to set different strengths for the tools, then you need to prompt the user for his/her specific strength for the tools, and you will need to select a specific strength for the AI choice.

****If you selected one non-default strength for both, then you will not be penalized****

You must have the proper constructors, destructors, and assignment operator overload for the Tool, Rock, Paper, Scissor, and RPSGame classes.

Your member variables in all classes must be private or protected for encapsulation rules.

You must have your class definitions in a .hpp file and your implemented classes in .cpp files.

You must also have your main function in a play_game.cpp file, separated from the class implementations.

Create a Makefile for your project. Finally, write a report to include your design decision, test plan & results and design changes.

What to submit:

- Your program files (.cpp .hpp).
- The reflections document (pdf), also include the work distribution in your group .
- A makefile.
- All the files must be submitted in a zip archive.

Grading:

- Programming style and documentation -10%
- Create the Tool class and object -10%
- Create the rock, paper, scissors class -15%
- Create the RPSGame class and implement the game flow - 40%
- Implement input validation functions -5%
- Reflection -20%