Dice Lab

In this lab we will build a dice game. We are going to build this project using multiple classes and files. I have provided some code below that you need to copy and paste into files that you will create, but there is not any skeleton code for this project – there is a solution video posted in Canvas. See the Program Grading Rubric for details on how your program will be scored. Points will be awarded as follows:

- Documentation & Style (10%)
  Program header
  Clear documentation between each sub-problem
- Syntax (50%)
- Correctness (10%)
- Logic (10%)
- Reusability (10%)
- Usability (10%)
- Total 30 Pts

In this lab we are going to create a dice game simulation. I don't know how much I need to explain it in text as I have published a solution video. Here is a brief summary of the game:

The game is a two-player dice game where the first player to reach 100 points wins. A human player alternates turns with a computer opponent. Each turn consists of rolling a six-sided die until the player decides to "hold" the dice, at which point the round total is added to the overall total points or a 1 is rolled and the turn is over with zero points added to the total score. A player can continue to "roll" the dice for more points. If a 1 is rolled then the player gets zero points for the round and the turn ends. Players alternate turns until a player reaches or exceeds 100 points and a winner is declared. If that does not make sense, then watch the solution video.

For this project you will create two classes and a total of 5 separate files. You will write the class definitions and the main. You will create the header files for the classes as well but all you need to do is paste the content from the header files that I have posted below. You will need to copy and paste these to files that are added to your project. The indentation probably won't turn out well, but I am not sure that I can fix that.

SEE NEXT PAGE

Player.h

```
#ifndef _PLAYER_H_
#define _PLAYER_H_

#include "Dice.h"

class Player {
private:
    int totalScore;//the total score for the game
    int roundScore;//the score for the round at the moment
    bool isCPU;//is this player a human or a CPU
public:
    Player();//default constructor
    Player(bool b);//one arg constructor - sets isCPU to true or false
    int getTotalScore();//get the total score for the player
    void resetRoundScore();//set the round score to zero
    int getRoundScore();//get the current round score
    void playRound();//check whether the player is human or CPU and call
the correct "Turn" function
    void playerTurn();//Manage a player round of player turns. Complicated
function. Continuously ask the player to roll or hold. Evaluate results.
    void cpuTurn();//Manage a CPU turn. Similar to a player turn but with
no input. The CPU will roll until they reach 20 points.
};

#endif
```

Dice.h

```
#ifndef _DICE_H_
#define _DICE_H_

#include <random>

class Dice {
private:
    int numSides;//number of sides on the die
    int value;//the side that is facing "up" on the die
public:
    Dice();//default constructor - will be overriden by a more useful one
arg constructor
    Dice(int x);//one arg constructor - this determines the number of
faces on the die
    void roll();//roll the dice. Generate a random number between 1 and
numSides and set dice value to that number
    int getValue();//get the current value of the die
};

#endif
```

SEE NEXT PAGE

This is the point where you can begin or start watching the solution video. If you want to give it a try independently, then I will try to explain how the Main(), PlayerTurn() and CPUTurn() should work. The rest of the functions are self-explanatory – that does not mean that they should not be documented. You have a solution available to you, but you need to document the classes and the functions. You are free to add any additional functionality that you want.

Main

Main is pretty simple. Create two players, one of type human and one of type CPU. You will create a loop that runs until one of the players reaches 100 points. You will need to create a lot of output here to make the game more playable. You should communicate the round number and a summary of the scores for each round. Once the game status is clear, then you will call the playRound() function for each player until the game ends.

playerTurn

This is the most complex function in the project. There is a lot of output that needs to be displayed. I want to remind you that namespaces don't belong in class definitions so you will have to qualify cout and endl with std::.

You will need a loop that runs until the player holds the dice. You will need to display a prompt and get the user input. The input will need to be validated. I validated it in this function, but you could write a function to validate the input if you wanted to – that would need to be added to the .h file.

You will need to track and display the total of the rolls for the round after each roll. If a 1 is rolled the turn is over and no points are added. If the player holds the dice, then the total for the round is added to the total score. Continue to roll until either a 1 is rolled, 100 total points are reached, or a hold is specified via input. When a player holds, the round score is added to the total score and the turn is over. After the turn, the results of the turn and the game should be clearly displayed.

cpuTurn

This is similar to the player turn, but there is no user input. Instead of waiting on user input, this function will loop until either a 1 is rolled or the CPU accumulates 20 points. After the turn, the results of the turn and the game should be clearly displayed.

When you are finished, then please submit your completed files through Canvas. You will have 5 files (Player.h, Player.cpp, Dice.h, Dice.cpp and main.cpp)