**Project 1**

A picture containing table

Description automatically generated

The project is to create a 2-player game of dice.

Game Rules:

* The game consists of two players (1 – even player & 2 – odd player). The game starts with one player at random.
* The user selects the number of rounds to be run. Each dice round consists of 3 values:
  + **Dice** – value of the dice rolled. It is a random number between 1-6.
  + **Points** – value associated with the round. It is a random number from 10-100 in multiples of 10. (10 \* random number between 1-10).
  + **Type** – the type of the round. There are 3 types of rounds:
    - **BONUS** – if the player dice roll is successful, the player gains 100 points. If unsuccessful, 100 points are deducted from the player’s score.
    - **DOUBLE** – if the player dice roll is successful, the player gains twice the points. If unsuccessful, those points are deducted from the player’s score.
    - **REGULAR** – if the player dice roll is successful, the player gains the points value decided above. If unsuccessful, those points are deducted from the player’s score.
    - Probability of the round types:
      * **20% for BONUS, 20% for DOUBLE and 60% for REGULAR**
* For each round, one of the following cases can occur:
  + Player-1 (even player) is the current player and the dice rolled is even   
    OR   
    Player-2 (odd player) is the current player and the dice rolled is odd
    - Current player gains the points, based on the type of the round (see above). The player’s turn continues in the next round.
  + Player-1 (even player) is the current player and the dice rolled is odd   
    OR   
    Player-2 (odd player) is the current player and the dice rolled is even
    - Current player incurs penalty of the same amount of points, based on the type of the round (see above). In the next round, the current player is changed to the other player.
* At the end of all rounds, the player with the highest points wins the game.

**Task 1 – create and write dicegame.h**

* Include the header guard in the correct format.
* Create an **enum** named **ROUNDTYPE** as shown below:  
  **typedef enum ROUNDTYPE   
  {   
   BONUS,   
   DOUBLE,   
   REGULAR   
  } ROUNDTYPE;**
* Create a **struct** named **DiceRound** which will contain 3 members: dice (random integer 1-6), points (random integer 10-100 in multiples of 10), and RoundType (enum – see above).
* Write all the function prototypes – see Task 2 below.

**Task 2 – create and write dicegame.c**

Write the following functions that use the struct and enum above, considering the game rules.

* **getRandomNumber()** – This function takes in 2 integer parameters min and max, computes a random number between min and max, **inclusive**, and returns it.
* **fillRounds()** – This function takes in 2 parameters: DiceRound dynamic array and size of the array. It fills the array of DiceRound by allocating all 3 of its members. ‘dice’ will be a random integer 1-6. ‘points’ will be a random integer between 10-100, in multiples of 10. ‘type’ will be random ROUNDTYPE based on the probability explained above. To assign these values, you can use the **getRandomNumber()** function. This function will not return anything.
* **getRoundPoints()** – This function will take in one parameter which is one DiceRound object from the entire array. It will use the ‘type’ of the round and the ‘points’ to determine the actual points for the round to be used. The decision is based on the round type explanation in the game rules above. This function will return a single integer value.
* **printRoundInfo()** – This function will take in 1 parameter which is a DiceRound object for the current round. It will not return anything, rather just print the values in the format:  
  Type : BONUS  
  DICE : 4  
  POINTS : 50
* **printPlayerInfo()** – This function will take in 2 integer parameters, which are the point values of both the players. It will not return anything, rather just print the values in the format:   
  P-1 : 20  
  P-2 : 10

**Task 3: Complete** **the main.c**

* Download the attached main.c
* Follow the instructions written in the comments in the main() function. The main() is the driver of the program. It calls the functions above to play the game.

**Rubric**

Be sure that your code follows the class coding style requirements. Your output should be similar in format as compared to the sample output attached. The rubrics document gives you detailed split of the code requirements.

**Submission**

Create a folder named your abc123, place all program files in this folder. Zip the folder and submit this abc123.zip file.