Design 1 Template: Go Hooping

Part 1: Understanding the Problem:
(1 pt) [Rephrase the problem statement here]

To ensure that you understand the problem correctly, answer the following guiding questions:

1. (2 pts) What are the three possible states for each ball shot, and what is the character notation for each state?

2. (2 pts) For non-money-ball rack positions, state if (a) to (d) is a valid output from random generation. If it is not valid, explain why:
   a. X X X X X
   b. X O O O O
   c. M O O O O
   d. X O X O M

3. (2 pts) For the following ball shooting outcomes, filling the blanks for each rack score, and the total score:
   a. Rack 1: O O O O M | ____ pts
      Rack 2: O O O O M | ____ pts
      Rack 3: M M M M M | ____ pts
      Rack 4: O O O O M | ____ pts
      Rack 5: O O O O M | ____ pts
   b. Rack 1: X O O O X | ____ pts
      Rack 2: O X O O M | ____ pts
      Rack 3: O X O X | ____ pts
      Rack 4: X O X O X | ____ pts
      Rack 5: M X M M X | ____ pts

   Total: __________
   Total: __________

4. (1 pt) For each execution, what inputs do you need from the user?

5. (1 pt) For each execution, what outputs do you need to print?

(1 pt) [List assumptions that you made]
Part 2: Program Design:
To help you break the problem down into smaller subtasks, answer the following guiding questions:
1. (1 pt) Without error handling, how would you let the user/player choose their “money-ball rack” position?

2. (1 pt) For a single ball shot, how would you generate the random number to meet the requirement of “50% chance of making the shot”?

3. (1 pt) For one rack position (5 ball shots), regardless of it being money-ball rack or not, how would you simulate the result?

4. (1 pt) What needs to be done so that your implementation could generate different results every time you run the program?

5. (1 pt) How would you use your design from question 3 above, and expand it to 5 racks?

6. (4 pts) How many arrays would you use to store the ball shooting result? Is it one dimension or two dimension? What is the size of your array? What is the data type of your array elements?

7. (1 pt) How would you print out the ball shooting result for 1 rack? 5 racks?
8. (1 pt) Before the scoring calculation, how would you represent your array in your program, so it can tell the difference of regular ball (1 point), missed (0 point), and money ball (2 points)?

9. (1 pt) How would you incorporate the “money-ball” rack chosen by the user into your program?

10. (2 pts) How would you calculate the score for 1 rack? 5 racks? What information needs to be stored in this step to help calculate the total score?

11. (1 pt) How would you modify your output functionality so that it contains the information of scoring?

12. (1 pt) How would you alternate the player, i.e., from Player 1 to Player 2?

13. (1 pt) How would your program determine the winner (or a tie game)?

14. (1 pt) How would you implement the “play again” functionality? What kind of loop are you going to use?
15. (2 pts) Now, considering error handling features, what changes do you need to make when taking user inputs? How would you validate the input and re-prompt until a valid one is provided?

(10 pts) [Use your answer above, create pseudocode (or flowchart) for each of the function that you plan to create]

Tip: In order to modularize your program, let each function handle one thing/task. When decompose a program into functions, try listing the verbs/tasks that are performed to solve the problem.

Part 3: Program Testing

To help you consider the possible test cases, answer the following guiding questions:

1. (1 pt) What is an example of good input when prompted for the money-ball rack?

2. (1 pt) How would you tell/test if the generated ball shooting result is valid?

3. (1 pt) How would you tell/test if the scores are correctly calculated?

4. (1 pt) Are there any inputs that would cause your program to crash? If so, what are they?

(6 pts) [Create a testing table that has representative good, bad, and edge cases, and their expected outputs]