MathCodeLab.Level1.Loops Puzzle-Based Programming Homework

Name: _____ **Date:** _____

Get ready for some exciting programming puzzles! Each problem combines loops, conditions, data types, and operators in fun challenges. Show your work and write clean, readable code.

🔐 Problem 1: Magic Number Lock (Constraints + Operators)

The Challenge:
A mysterious treasure chest has a special lock! The chest opens only if you enter a number that follows this magic rule:

🔑 **Magic Rule:** The number must be divisible by **3 AND 5**, but **NOT by 2**.

Your Mission:
Write a program that:

1. Asks the user to enter a number
2. Checks if the number follows the magic rule
3. Tells the user if the treasure chest opens or stays locked

Example:

- Input: `15` → Output: "🎉 Treasure chest opens! 15 follows the magic rule!"
- Input: `30` → Output: "❌ Chest stays locked! 30 is divisible by 2."
- Input: `7` → Output: "❌ Chest stays locked! 7 doesn't follow the magic rule."

Hints:

- Use the `%` (modulo) operator to check divisibility
- A number is divisible by another if `number % divisor == 0`
- Combine conditions with `and` and `not`

Write your code here:

```
# Your Magic Number Lock program
```

Test your program with these numbers: 15, 30, 45, 21, 75, 90

⭐ Problem 2: Star Pyramid Challenge (Loops)

The Challenge:
Build a beautiful star pyramid! Your program should create a pyramid pattern using stars (`*`).

Your Mission:
Write a program that:

1. Asks the user for the height of the pyramid (number of rows)
2. Prints a star pyramid with that many rows

Example: For input `n = 5`:

```
*
**
***
****
*****
```

Example: For input `n = 3`:

```
*
**
***
```

Hints:

- Use a `for` loop to control the number of rows
- For each row, print the correct number of stars
- Row 1 has 1 star, row 2 has 2 stars, etc.

Write your code here:

```
# Your Star Pyramid program
```

🌟 **Bonus Challenge:** Can you make a reverse pyramid or a diamond shape?

🔍 Problem 3: Guess the Data Type (DataTypes)

The Challenge:
Create a smart program that can identify what type of data the user enters!

Your Mission:
Write a program that:

1. Asks the user to enter something (anything!)
2. Analyzes what they entered
3. Tells them if it's an **integer**, **float**, or **string**

Examples:

- Input: `"42"` → Output: "That's an integer! 🔢"
- Input: `"3.14"` → Output: "That's a float! 🔢.🔢"
- Input: `"Hello"` → Output: "That's a string! 📝"
- Input: `"123.0"` → Output: "That's a float! 🔢.🔢"

Hints:

- Get input as a string first
- Try to convert it to different types using `int()` and `float()`
- Use try-except blocks to catch conversion errors
- Or use string methods like `.isdigit()` and check for decimal points

Write your code here:

```
# Your Data Type Detective program
```

Test with these inputs: 42, 3.14, Hello, 123.0, -57, 0

🤖 Problem 4: Robot Steps Puzzle (Loops + Constraints)

The Challenge:
A friendly robot starts at position 0 and follows special movement rules!

🤖 Robot Rules:

- If the current step number is **odd**: robot moves **+3 steps forward**
- If the current step number is **even**: robot moves **+2 steps forward**

Your Mission:
Write a program that:

1. Asks the user how many moves the robot should make
2. Simulates each move following the robot rules
3. Shows the robot's position after each move
4. Prints the final position

Example: If user enters `n = 5`:

```
Move 1 (odd): +3 steps → Position: 3
Move 2 (even): +2 steps → Position: 5
Move 3 (odd): +3 steps → Position: 8
Move 4 (even): +2 steps → Position: 10
Move 5 (odd): +3 steps → Position: 13

🤖 Robot's final position: 13
```

Hints:

- Use a `for` loop to count the moves
- Check if the move number is odd or even using `% 2`
- Keep track of the robot's position in a variable
- Update the position after each move

Write your code here:

```
# Your Robot Steps program
```

Test with these values: n = 3, n = 6, n = 10

🎯 Reflection Questions

After completing all puzzles, answer these questions:

1. **Which puzzle was the most challenging and why?**

   ___

2. **What's one new thing you learned about loops from these puzzles?**

   ___

3. **How did using the modulo operator (%) help you solve problems?**

   ___

4. **Which puzzle would you like to extend or modify? How?**

   ___

🏆 **Bonus Challenges** (Optional)

If you finish early, try these extensions:

1. **Magic Lock Extension**: Find all magic numbers between 1 and 100
2. **Pyramid Power**: Create a diamond pattern or a hollow pyramid
3. **Data Type Pro**: Handle negative numbers and scientific notation
4. **Robot Adventure**: Make the robot move backward sometimes, or add obstacles

---

💡 **Remember:**

- Test your programs with different inputs
- Use meaningful variable names
- Add comments to explain your logic
- Have fun with these puzzles!

Good luck, young programmer! 🚀