Number Guessing Game Deluxe

This is an engaging and interactive project that goes beyond simple static calculations. It creates a dynamic "game loop" that reacts to user input in real-time until a win or loss condition is met.

Concept: A classic console-based puzzle game where the player must use logic and intuition to find a hidden number generated by the computer.

Core Features:

- Multiple Difficulty Modes: Offers an "Easy Mode" (Range 1-50, unlimited tries) for casual play and a "Hard Mode" (Range 1-100, 7 attempt limit) for a challenge.
- Smart Feedback System: Provides real-time "Too High" or "Too Low" hints after every guess to guide the player.
- Random Number Generation: Utilizes srand(time(0)) to ensure every session has unique secret numbers.
- Win/Loss Logic: Tracks attempt counts in Hard Mode to trigger a "Game Over" if the player runs out of guesses.

Why it's a good minor project: It demonstrates mastery of essential C constructs like composite loops (do-while nested within do-while), complex conditional logic, and integrating standard libraries (time.h, stdlib.h) for dynamic behaviour.

How it works?

1. **Analyze the Request:** The user wants a brief description of how the previously provided "Number Guessing Game Deluxe" code works.
2. **Identify Key Components of the Code:**
    - Imports: stdio.h (I/O), stdlib.h (rand/srand), time.h (seeding).
    - main() function:
        - Seeding random number generator: srand(time(0)).
        - Menu loop: do-while loop for playing again or switching modes.
        - Switch statement: Handles user choice (Easy/Hard/Exit) and calls playGame.
    - playGame() function (core logic):
        - Arguments: difficultyRange (max number), maxAttempts (limit, 0 for unlimited).
        - Secret Number Generation: rand() % difficultyRange + 1.
        - Game Loop: do-while loop for user guesses.
        - Feedback Logic: if/else if/else to check if guess is correct, high, or low.
        - Win/Loss Logic: Checks if matched (win) or if attempts exhausted (loss in hard mode).
3. **Draft the Description (Iterative Refinement for Brevity and Clarity):**
    - *Draft 1 (Mental Outline):* The code starts by importing libraries. In main, it seeds the random generator so numbers are different each time. Then it enters a loop showing a menu. user picks easy or hard. This calls playGame with different settings. PlayGame picks a random number. It loops, asking for guesses. It tells you if you are high or low. It counts attempts. If you run out of attempts in hard mode, you lose. If you guess right, you win.

- *Draft 2 (More Technical/Precise):* The program uses standard C libraries for input/output and random number generation. The main function initializes the random seed using the current time to ensure unique games. It presents a menu inside a do-while loop, allowing repeated play. selecting a difficulty calls playGame with specific parameters (range and attempt limit). playGame generates a secret number using modulo arithmetic. It enters another loop where the user guesses. Conditionals provide "Too High"/"Too Low" feedback. A specific check determines if the user has exceeded the maximum attempts in hard mode, ending the loop early if true.
- *Final Polish (Structuring for the user):* Use headings for clarity. Break down into "Setup," "Main Menu," and "Game Logic."

4. **Final Review against constraints:** Concise? Yes. Straightforward language? Yes. Accurate? Yes. Citations needed? No (explaining previously generated code, not source docs).

(Self-Correction during drafting): Ensure the explanation of srand(time(0)) is simple—it's crucial for *true* randomness but can be confusing if over-explained. Just say it ensures unique games. Explicitly mention how Easy vs. Hard mode is handled (passing different arguments to the same function).

5. **Final Output Generation** (matching the drafted structure).Here is a brief description of how the code works:

1. Setup and Initialization

- **Libraries:** The code includes <stdio.h> for input/output (like printf, scanf), <stdlib.h> for random number generation (rand, srand), and <time.h> to seed the random generator.
- **Seeding:** srand(time(0)) is called once at the start of main. This uses the current time to ensure the random numbers are different every time you run the program.

2. The Main Menu Loop

- The main function uses a do-while loop to keep the program running until the user chooses to exit (option 0).

- It displays a menu and uses a switch statement to handle the user's choice.
- Based on the choice, it calls the playGame function with different arguments:
    - **Easy Mode:** Calls playGame(50, 0) (Range 1-50, 0 means unlimited attempts).
    - **Hard Mode:** Calls playGame(100, 7) (Range 1-100, limit of 7 attempts).

3. Core Game Logic (playGame function)

- **Secret Number:** It generates a random number using (rand() % difficultyRange) + 1.
- **Guessing Loop:** A do-while loop repeatedly asks the user for a guess.
- **Feedback:** Inside the loop, if/else statements compare the guess to the secret number and print "Too High," "Too Low," or a success message.
- **Game Over Check:** It tracks the number of attempts. If maxAttempts is greater than 0 (Hard Mode) AND the user reaches that limit without guessing correctly, it forces the loop to end and reveals the number.