Part 1: yield return Exercises

Exercise 1: The Even Number Stream

Goal: Create a method that generates even numbers up to a limit.

- **Task:** Write a method GetEvens(int max) that uses yield return to provide only even numbers.
- **Constraint:** Do not use a List or Array inside the method.

C#

```
public IEnumerable<int> GetEvens(int max)
{
    // Your code here
}
```

Exercise 2: The Name Formatter

Goal: Transform data on the fly.

- **Task:** Given a list of names, create an iterator that yields each name in ALL CAPS with an exclamation mark (e.g., "Alice" becomes "ALICE!").
- **Context:** This demonstrates how yield can process data without creating a second collection in memory.

Exercise 3: The Infinite Counter

Goal: Understand that yield doesn't need a finish line.

- **Task:** Create a method InfiniteCounter() that starts at 1 and increments forever using an infinite while(true) loop.
- **Test:** In your Main method, use a foreach loop with this method but use a break inside the foreach after 10 iterations so your program doesn't crash!

Part 2: yield break Exercises

Exercise 4: The Early Exit Filter

Goal: Stop a sequence when a "Stop" command is found.

- **Task:** Write a method that iterates through a list of strings. If it encounters the string "END", use yield break. Otherwise, yield return the string.
- **Input:** ["Apple", "Banana", "END", "Cherry"]
- **Expected Output:** Only "Apple" and "Banana".

Exercise 5: The Range Guard

Goal: Prevent invalid processing.

- **Task:** Create a method TakeFirstN(List<int> numbers, int n).
- **Logic:** Loop through the numbers. If your counter reaches n, or if a number is negative, use yield break.
- **Why?** This simulates a "Safe" data reader.

Exercise 6: The Password Validator

Goal: Combine logic with termination.

- **Task:** Create an iterator that yields characters of a password one by one.
- **Logic:** If a space character ' ' is detected, use yield break because spaces are invalid, and we should stop reading the password immediately.