Generate Parantheses

To solve the Generate Parantheses problem, we backtracking to explore all valid combinations of well-formed parentheses.

Key Idea: At any point:
- you can add '(' if you still have some left to place.
- you can add ')' only if it won't exceed the number of '(' already placed.

This ensures that no invalid prefix is ever generated (like ")("), so we never need to check if a sequence is valid after.

Backtracking Strategy:
1. Start with an empty string.
2. Track: - open: number of '(' used so far.
   - close: number of ')' used so far.
3. At each step: - If open < n → add '('
   - If close < n → add ')'
4. If both open == n and close == n, you've built a valid sequence → add to results.

Time Complexity: - Roughly $O(2^n)$ for generating all combinations.
- For n = 8, this is manegeable.

Example: for n = 3:
Output: ["((()))", "(()())", "(())()", "()(())", "()()()"]