→ Advanced Recursion and Backtracking

Q) Generate all valid strings of given n.

eg: n=2 ⟶ output = { '(())', '()()' }

eg: n=3 ⟶ output = { '((()))', '(()())', '(())()', '()(())', '()()()' }

M→ → Global vector to store generated valid strings

```
VECTOR<STRING> VALID;


VOID GENERATE(STRING &S, INT OPEN , INT CLOSE)
{
      → Base case => If we have finished all our
};    brackets
      IF (OPEN == 0 && CLOSE == 0)
```

```
{
    -> Pushing a valid generated string in our
       global valid vector.
    VALID.PB(S);
    RETURN;
}

-> If we have opening brackets present
IF (OPEN > 0)
{
    S.PB('(');
    GENERATE(S, OPEN -1, CLOSE);
    -> Backtracking
    S.POB();
}
```

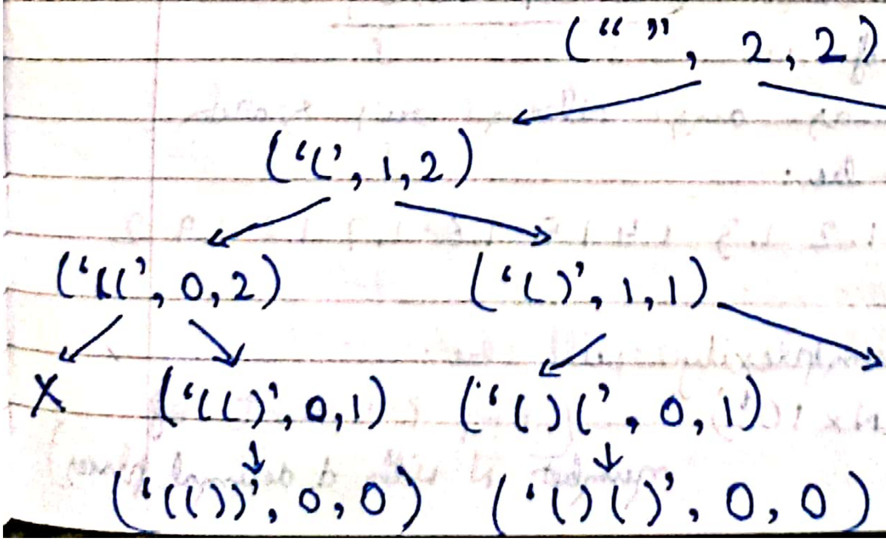
~~-> If we have more opening brackets in string than closing~~

```
-> If we have closing brackets present
IF (CLOSE > 0)
{
    -> If we have more opening brackets
       in string than closing
    IF (OPEN < CLOSE)
    {
        S.PB(')');
        GENERATE(S, OPEN, CLOSE -1);
        -> Backtracking
        S.POB();
    }
```

```
        }
}

INT MAIN ( )
{
    INT n;
    CIN >> n;
    STRING S;

    -> Generating string recursive function
    GENERATE (S, n, n);

    -> Printing the generated strings from global vector
    FOR (AUTO &ST : VALID)
    {
        COUT << ST << "\n";
    }
    RETURN 0;
}
```

-> It's recursion tree:

For n=2



NOTE: Monotonic functions : A function which maintains a given order.

e.g: Ds 1 2 3 4 ↑ seq.
7 6 3 2 ↓ seq.

Binary search works on Monotonic functions only.