10th June 2025

Problem 1: Maximum Meetings in one Room

⇒ Approach:

1. Pair each meeting's start and end time.
2. Sort the meetings by their end time.
3. Initialize the end time of last selected meeting to 0.
4. Iterate through sorted meetings if start time of current meeting is greater than end time of last selected one, count it and update the end time.

Pseudo code:

```
Meeting[] meetings = new meeting[n];
for (int i=0; i<n; i++)
{
  meeting[i] = new Meeting (start[i], end[i]);
}
Arrays.sort (meetings, (a,b) -> a.end - b.end);
int count = 0;
int lastEndTime = 0;
for(Meeting m : meetings){
    if( m.start >= lastEndTime){
      count++;
      lastEndTime = m.end;
    }
}   return count;
}
```

Dry run: n = 6

start = [1, 3, 0, 5, 8, 5]

end = [2, 4, 6, 7, 9, 9]

=> • Pair start & endTimes

(1,2) (3,4) (0,6) (5,7), (8,9), (5,9)

• Sort by end time:

sorted meetings = [(1,2) (3,4) (0,6) (5,7) (8,9) (5,9)]

• Initialize

count = 0

lastendTime = 0

(1,2) 1 >= 0 → count = 1, lastTimeEnd = 2

(3,4) 3 >= 2 → count = 2, " = 4

(0,6) 0 >= 4 × → skip

(5,7) 5 >= 4 → count = 3, lastTimeEnd = 7

(8,9) 8 >= 7 → count = 4, lastEndTime = 9

(5,9) 5 >= 9 × skip

∴ Total count = 4

problem 2: Number of ways to evaluate expression to True

⇒ Approaches:

1. Recursively divide the expression at each operator.
2. Count ways for left and right sub-expressions to be true and false.
3. Combine them using current operator and apply boolean logic.
4. Memorize result for overlapping subproblems.

pseudo-code:-

```
static Map<String, Integer> memo =
    new HashMap<>();
static int countWays(String s, int i, int j,
    boolean isTrue){
    if(i>j) return 0;
    if(i==j) return (isTrue==(s.charAt(i)=='T'))?
                1:0;
String key = i + "_" + j + "_" + isTrue;
    if(memo.containKey(key)) return memo.get(key);
int ways=0
for(int k= i+1; k<j; k+=2){
    char op = s.charAt(k);
    int lt = countWays(s, i, k-1, true);
int lf = countWays(s,i,k-1, false);
int rt = countWays(s,i, k-1, false);
int rf = countWays(s,k+1, j, true);
int rf = countWays(s,k+1, j, false);
if(op=='&') ways += isTrue ? lt*rt : lt*lf + lf*rt + lf*rf
    else if(op=='|') ways += isTrue ? lt*rt + lt*lf + lf*rt
}
memo.put(key, ways);
return ways;
```

Dry Run! S = "T|F&T^T"

- ((T|(F & (T^T)))
- ((T|((F&T)^T)))
- (((T|F) & (T^T)))
- (((T|(F&T))^T))
- (((T|F) & T)^T))

} output = 5