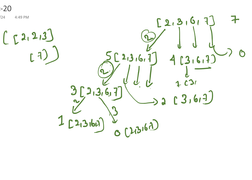
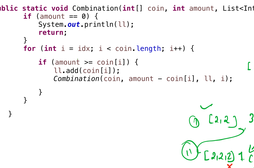
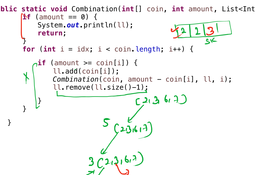
Lec-20

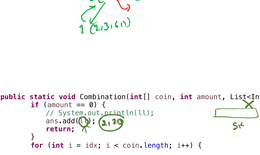


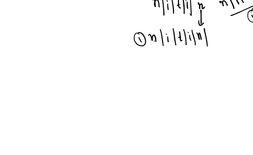
```
public static void Combination(int[] coin, int amount, List<Integer> ll, int idx) {
    if (amount == 0) {
        System.out.println(ll);
        return;
    }
    for (int i = idx; i < coin.length; i++) {
        if (amount >= coin[i]) {
            ll.add(coin[i]);
            Combination(coin, amount - coin[i], ll, i);
            ll.remove(ll.size()-1);
        }
    }
}
```

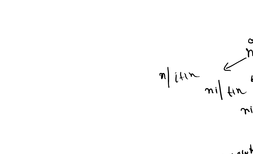


```
public static void Combination(int[] coin, int amount, List<Integer> ll, int idx) {
    if (amount == 0) {
        // System.out.println(ll);
        ans.add(ll); 3,3,6
        return;
    }
    for (int i = idx; i < coin.length; i++) {
        if (amount >= coin[i]) {
            ll.add(coin[i]);
            Combination(coin, amount - coin[i], ll, ans, i);
            ll.remove(ll.size() - 1);
        }
    }
}
```

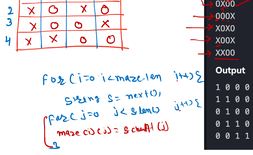


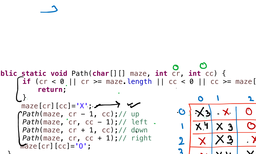




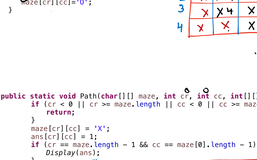


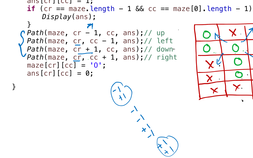
```
public static void Partitioning(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 1; i <= ques.length(); i++) {
        String s = ques.substring(0, i);
        Partitioning(ques.substring(i), ans + s + "|");
    }
}
```





```
public static void Path(char[][] maze, int cr, int cc) {
    if (cr < 0 || cc >= maze.length || cc < 0 || cc >= maze[0].length || maze[cr][cc]=='X') {
        return;
    }
    maze[cr][cc]='X';
    Path(maze, cr - 1, cc);// up
    Path(maze, cr, cc - 1);// left
    Path(maze, cr + 1, cc);// down
    Path(maze, cr, cc + 1);// right
    maze[cr][cc]='0';
}
```





```
public static void Path(char[][] maze, int cr, int cc, int[][] ans) {
    if (cr < 0 || cr >= maze.length || cc < 0 || cc >= maze[0].length || maze[cr][cc] == 'X') {
        return;
    }
    maze[cr][cc] = 'X';
    ans[cr][cc] = 1;
    if (cr == maze.length - 1 && cc == maze[0].length - 1) {
        Display(ans);
    }
    Path(maze, cr - 1, cc, ans);// up
    Path(maze, cr, cc - 1, ans);// left
    Path(maze, cr + 1, cc, ans);// down
    Path(maze, cr, cc + 1, ans);// right
    maze[cr][cc] = '0';
    ans[cr][cc] = 0;
}
```