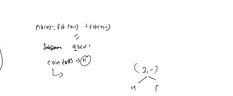
① FWD
Fwd

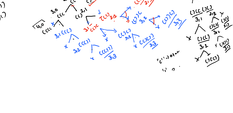


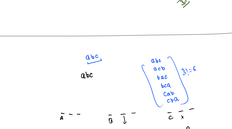
Given n pairs of parentheses, write a function to generate *all combinations of well-formed parentheses*.

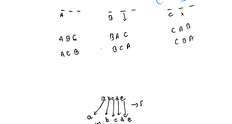
Example 1:

```
Input: n = 3
Output: ["((()))","((())()","(()()()","(()())()","()(()()"]
```

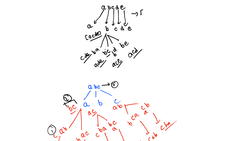
```
public static void Parenthesesi(int n, String ans, int open, int closed) {
    if (open == n && closed == n) {
        System.out.println(ans);
        return;
    }
    if (open < n || closed < open) {
        return;
    }

    Parenthesesi(n, ans + "(", open + 1, closed);
    Parenthesesi(n, ans + ")", open, closed + 1);
}
```



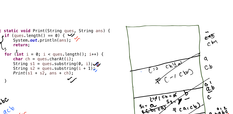




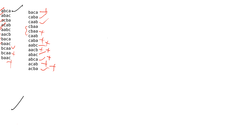
```
public static void Print(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < ques.length(); i++) {
        char ch = ques.charAt(i);
        String l = ques.substring(0, i);
        String r = ques.substring(i + 1);
        Print(l + r, ans + ch);
    }
}
```

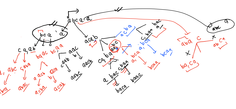


```
public static void Print(String ques, String ans) {
    if (ques.length() == 0) {
        System.out.println(ans);
        return;
    }
    for (int i = 0; i < ques.length(); i++) {
        char ch = ques.charAt(i);
        String l = ques.substring(0, i);
        String r = ques.substring(i + 1);
        Print(l + r, ans + ch);
    }
}
```



```
public static void KeyPad(String ques, String ans,List<String> ll) {
    if (ques.length() == 0) {
        System.out.println(ans);
        ll.add(ans);
        return;
    }

    char ch = ques.charAt(0);
    String ant = ques.substring(1);
    for (int i = 0; i < keypad[ch - '2'].length(); i++) {
        KeyPad(ant, ans + keypad[ch - '2'].charAt(i),ll);
    }
}
```





```
// mouse row | mouse col | cheese row | cheese column | curr | curr | curr | curr
public static void Ratinamaze(int mr, int mc, int cr, int cc, String ans) {
    if (mr == cr && mc == cc) {
        System.out.println(ans);
        return;
    }
    if (cr < mr || cc < mc) {
        return;
    }

    Ratinamaze(mr + 1, cr, mc, ans + "D");
    Ratinamaze(mr, cr + 1, mc, ans + "R");
}
```