Greedy Algorithms Contd...

* Minimum number of coins to reach a particular value/amount.

List<Integer> coins = { 1, 2, 5, 10, 20, 50, 100, 200, 500, 1000, 2000 };

{ 50, 20, 20, 1 }

int V_1 = 91;

int V_2 = 31;

int V_3 = 10;

91 − 50 = 41
41 − 20 = 21
21 − 20 = 1
1 − 1 = 0

31 − 20 = 11
11 − 10 = 1
1 − 1 = 0

```
List<Integer> result = new ArrayList<>();
for (i = coins.size() - 1; i >= 0; i--) {
    while ( V >= coins.get(i) ) {
        V -= coins.get(i);
        result.add(coins.get(i));
    }
}
no of coins -> res.size();
```

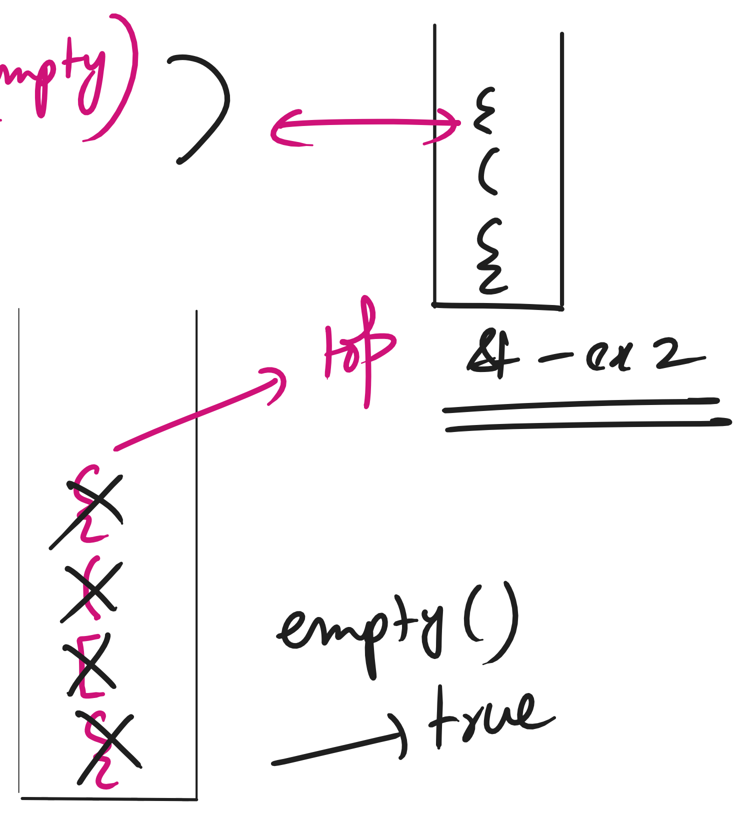
Balanced Parentheses :

String str1 = " {[({})]} " true

String str2 = " {({})]] " false

'{' (Stack : LIFO property)

} (empty))



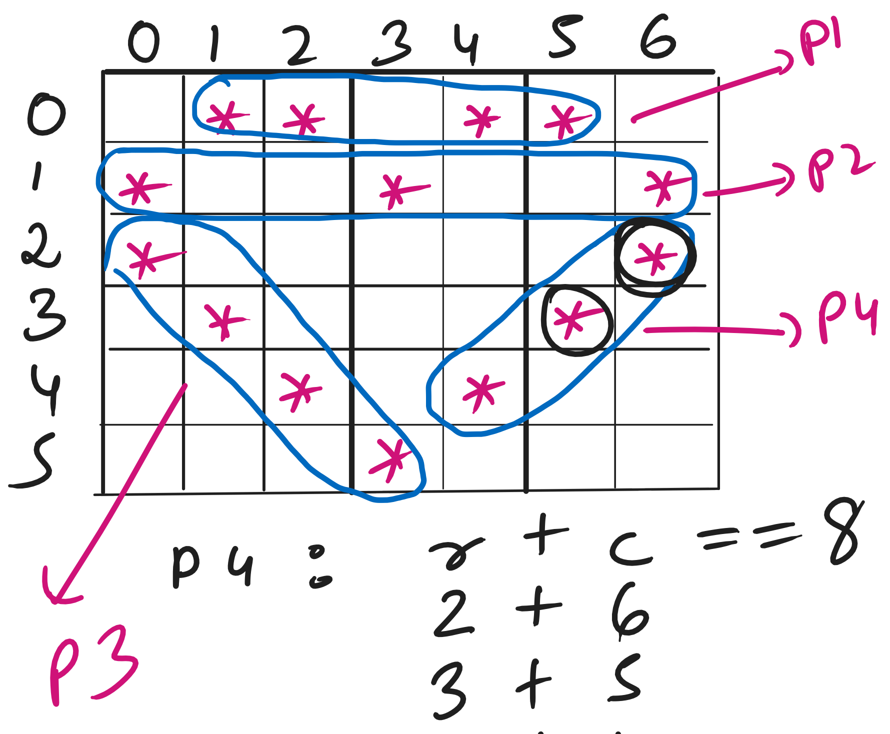

```
if ( str.charAt(i) == '{' || '[' || '(' )
    st.push( str.charAt(i) );
else if ( str.charAt(i) == '}' &&
    st.top() == '{'
    st.pop()
```

']' && '[' ')' && '('

TCS → Basic 3.5 (5 mins) NQT 4.5 – 6 (4 mins) Ninja 7 – 8 (3 mins) LPA

(Static Pattern)

(Break into parts)



P1 : r == 0 && c % 3 != 0
c 1, 2, 4, 5

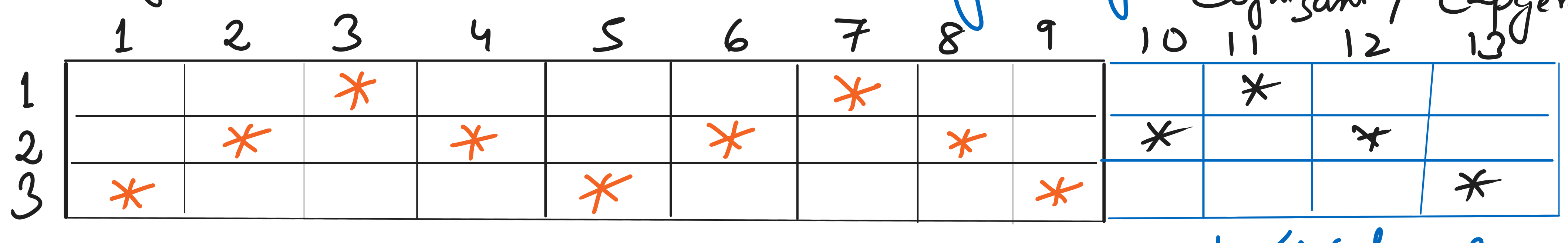
P2 : r == 1 && c % 3 == 0
c = 0, 3, 6

P3 : r − c == 2 if '*'
2 − 0
3 − 1
4 − 2
5 − 3
'#' ',' else " "

P4 : r + c == 8
2 + 6
3 + 5
4 + 4

Dynamic Patterns :→

Zig Zag Pattern (2 mins)
Cognizant / Capgemini



row (constant) = 3
col (variable) = 9, 13, 17, 21 and so on

(Maths)

r1 3, 7, 11 ⇒ c % 4 == 3
r2 even no ⇒ c % 4 == 2
r3 1, 5, 9, 13 ⇒ c % 4 == 1

(Package = folder) Access Modifier Table

Modifier Name	Inside Class	Outside Class	Inside Package	Outside Package
public	Yes	Yes	Yes	Yes
private	Yes	No	No	No
protected	Yes	Yes	Yes	Yes extends (Inheritance)
default	Yes	Yes	Yes	No

default ↓ (p² = package private)