Recursion (Google, Amazon Questions)

★ LeetCode - 17. Letter Combinations of a Phone Number

Take example like this

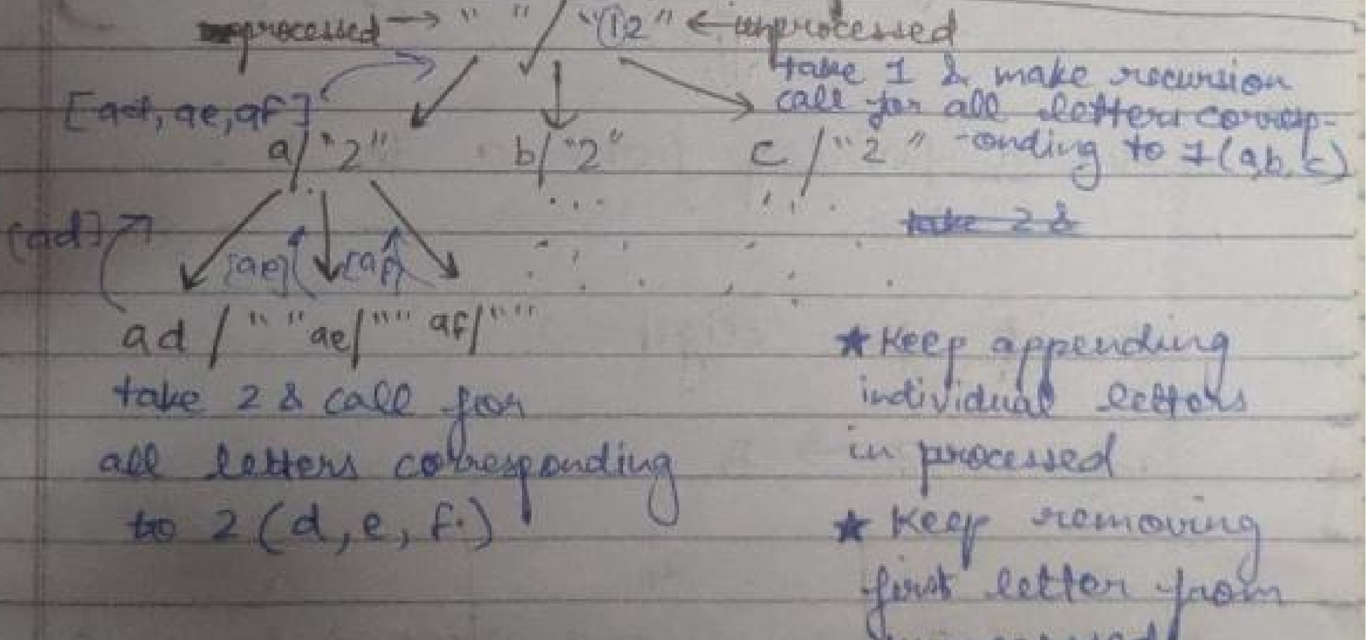
1	2	3
abc	def	ghi
0 1 2	3 4 5	6 7 8
4	5	6
jkl	mno	pqr
9 10 11	12 13 14	15 16 17
7	8	9
stu	vwx	yz
18 19 20	21 22 23	24 25

" 1 2 "

ad ← choose d for 2
↑
choose a for 1

ae ← choose e for 2
↑
choose a for 1

af ← choose f for 2
↑
choose a for 1



processed → " " / "12" ← unprocessed
take 1 & make recursion call for all letters corresponding to 1 (a,b,c)
[ad, ae, af]
a/"2"
b/"2"
c/"2"
~~take 2 &~~
[ad]
[ae]
[af]
ad/"" ae/"" af/""

take 2 & call for all letters corresponding to 2 (d, e, f.)

★ Keep appending individual letters in processed

★ Keep removing first letter from unprocessed

★ When unprocessed is empty you have found an answer ⇒ return it.

* Converting digits to corresponding alphabets

1	2	3
a b c	d e f	g h i
0 1 2	3 4 5	6 7 8
4	5	6
j k l	m n o	p q r
9 10 11	12 13 14	15 16 17
7	8	9
s t u	v w x	y z
18 19 20	21 22 23	24 25 ⟹ for idx=26 skip it

· For each digit the range can be defined like this

$$\text{digit range} = \Big[\underbrace{(\text{digit}-1) * 3}_{a},\ \underbrace{\text{digit} * 3}_{b}\Big)$$

⇓

run a for loop from a to b-1

~~fo~~ ~~it~~ for digit = 2

index = 3 → 5

3	4	5
first char of 2	2nd char of 2	third char of 2
'a' + 3 = 'd'	'a' + 4 = 'e'	'a' + 5 = 'f'

↑ char to add in processed

Date: / /
Page No:

printing all combinations

```
static void pad(String p, String up){
   if(up.isEmpty()) {
        System.out.println(p);
        return;
   }

   int digit = up.charAt(0) - '0'; // convert '2' to 2

   for(int i=(digit-1)*3; i<digit*3; i++) {
        if(i>25) continue;
        char ch= (char)('a'+i);
        pad(p+ch, up.substring(1));
   }
}
```

Date:
Page No.

storing answers in Arraylist

```
static ArrayList<String> padRet( String, p, String up){

    if(up isEmpty()) {
Return arraylist containing Answer
      > ArrayList<String> list = new ArrayList<>();
        list.add(p);
        return list;
    }

    int digit = up.charAt(0);
Make Arraylist for All answers
     > ArrayList< String> list = new ArrayList<>();

    for(int i=(digit-1)*3; i< digit*3; i++){   <- if(i>25) continue;
Add answers  char ch = (char) ('a'+ i);
from
all the    > list.addAll( padRet( p+ch, up.substring(1)));
calls    }
                 <- return final list
    return list;

}
```

returning count of all answers

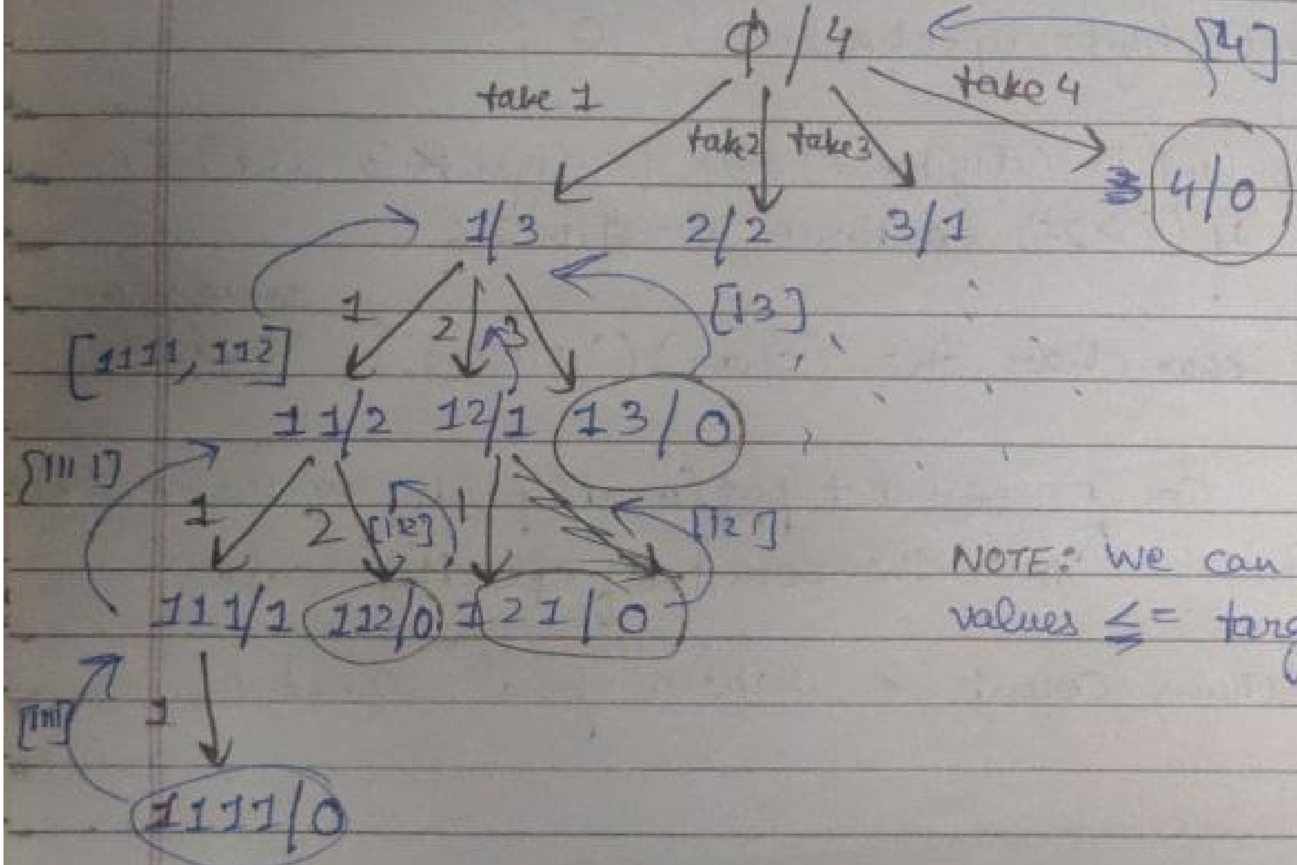
```
static int padCount(String p, String up) {

 if(up.isEmpty) {
        return 1;  <- 1 answer found
                      return 1
 }

 int count = 0;  <- Counting answers
 int digit = up.charAt(0) - '0';

 for(int i = (digit - 1) * 3; i < digit * 3 ; i++) {
    if (i > 25) continue;  <- skip 26
                                              conversion
    ~~char~~ char ch = (char)('a' + i);  <- to char

     Count = count + padCount(p+ch, up.substring(1));
 }           ↖ ~~Add the count of all the calls~~

 return Count;  <- return final answer
}
```

Dice Combination

Get 4 1, 2, 3, 4, 5, 6

Answers [(4), (2,2), (1,1,2), (3,1)]

* ~~Here~~ ~~also~~ we are ~~getting~~ taking something & removing something.



NOTE: We can take values ≤= target

print all answers

```
static void dice(String p, int target){
  if(target==0){
      System.out.println(p);
  }   return;

  for(int i=1;i<=6 && i<=target; i++){

      dice(p+i, target-i);
  }
}
```

returning List stored with all combinations

```
static ArrayList<String> diceRet(String p, int target){
 if(target==0){
    ArrayList<String> list = new ArrayList<>();
    list.add(p);
 }

 ArrayList<String> list = new ArrayList<>();

 for(int i=1; i<=6 && i<=target; i++){

    list.addAll(diceRet(p+i, target-i));
 }
 return list;
}
```