n_1 <— S_1 → send

n_2 ← S_2 → + more

n_3 S_3 → money

s

e

n

d

m

o

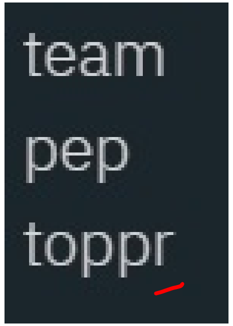
r

y

(0, 1, 2, ... 9)

$$\begin{array}{r} 1\ 1\ \ \ \\ 0942 \\ +\ \ 393 \\ \hline 01335 \end{array}$$

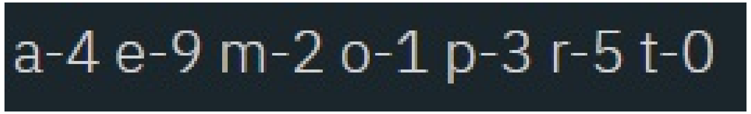
✓



team → s_1

pep → s_2

toppr → s_3



a-4 e-9 m-2 o-1 p-3 r-5 t-0

t → 0

e → 9

a → 4

m → 2

p → 3

o → 1

r → 5

Task

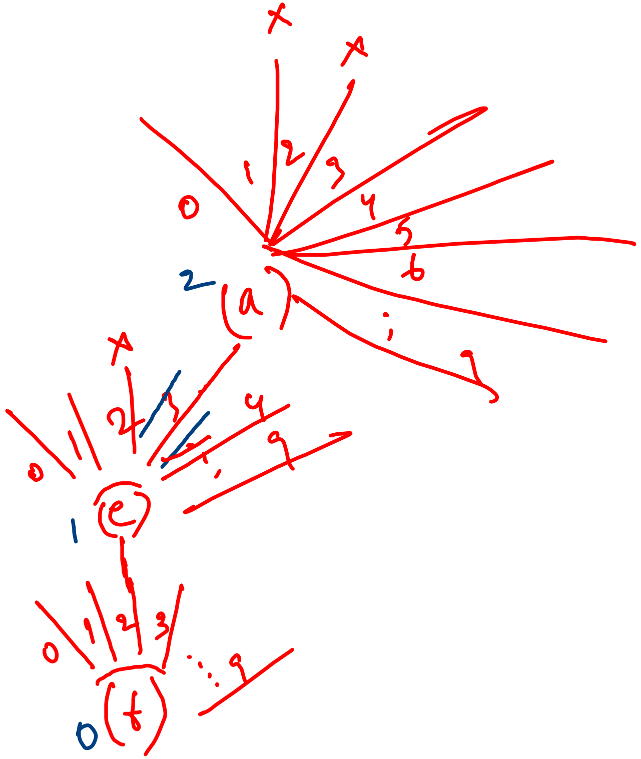
→ find distinct mapping for each character → digit which satisfies $n(s_1) + n(s_2) = n(s_3)$

$\leftarrow S_1$

$\leftarrow S_2$

S_7

$S_1 + S_2 \ldots S_7$



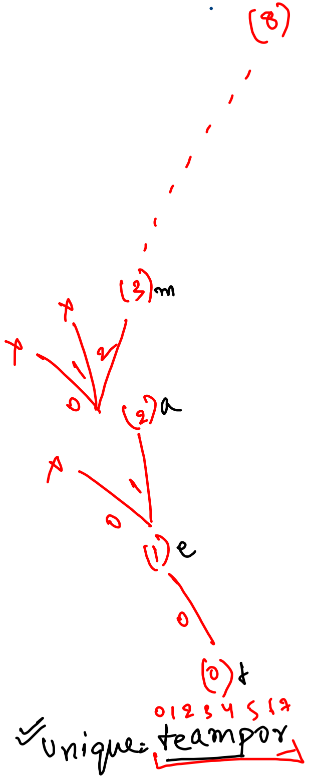
CharIntMap

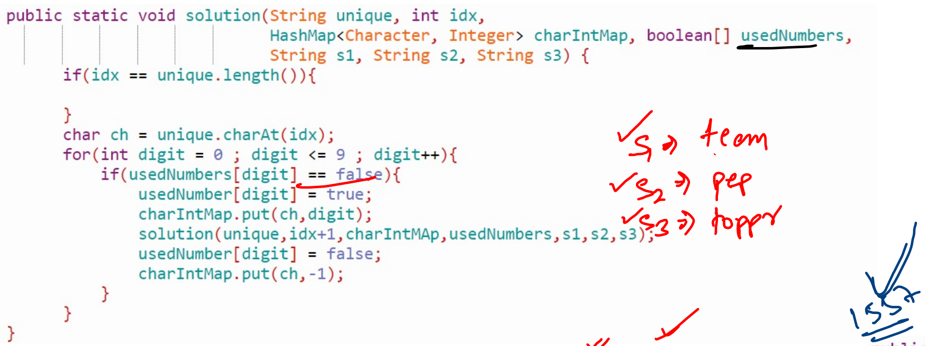
t	~~-1~~ 2
e	~~-1~~ ~~3~~ -1
a	-1
m	-1
p	-1
o	-1
r	-1

used →

0	1	2	3	4	5	6	7	8	9
F	F	~~F~~ T	~~F~~ ~~T~~ F	F	F	F	F	F	F

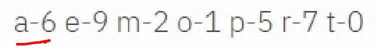
Unique: teampor



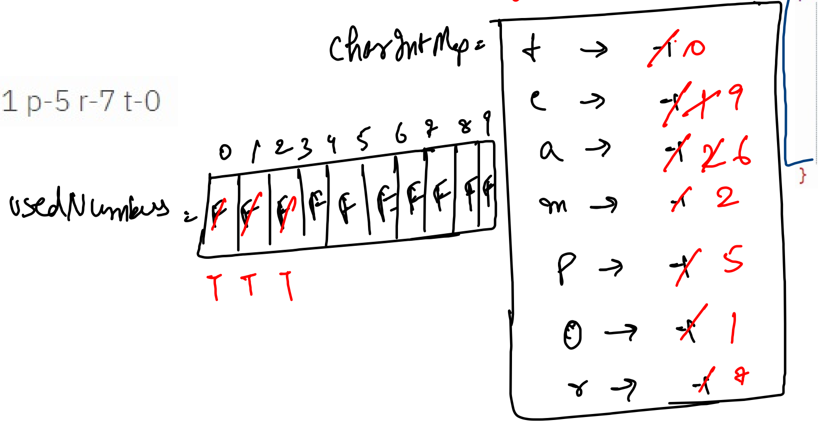


```java
public static void solution(String unique, int idx,
                            HashMap<Character, Integer> charIntMap, boolean[] usedNumbers,
                            String s1, String s2, String s3) {
    if(idx == unique.length()){

    }
    char ch = unique.charAt(idx);
    for(int digit = 0 ; digit <= 9 ; digit++){
        if(usedNumbers[digit] == false){
            usedNumber[digit] = true;
            charIntMap.put(ch,digit);
            solution(unique,idx+1,charIntMAp,usedNumbers,s1,s2,s3);
            usedNumber[digit] = false;
            charIntMap.put(ch,-1);
        }
    }
}
```

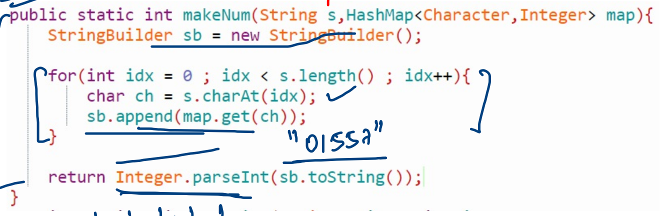
✓s1 → team
✓s2 → pep
✓s3 → topper



```
a-6 e-9 m-2 o-1 p-5 r-7 t-0
```

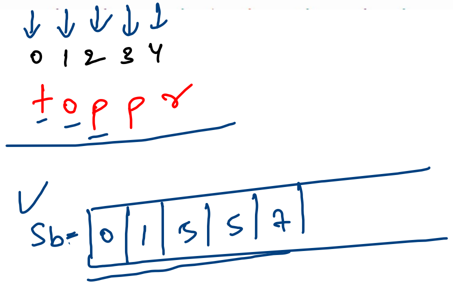


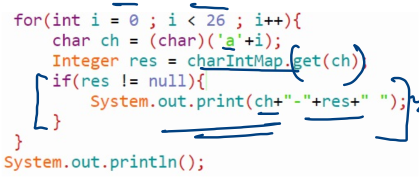
1557



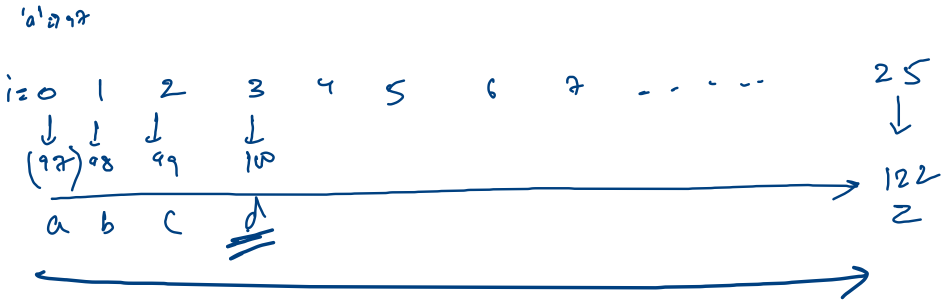
```java
public static int makeNum(String s,HashMap<Character,Integer> map){
    StringBuilder sb = new StringBuilder();

    for(int idx = 0 ; idx < s.length() ; idx++){
        char ch = s.charAt(idx);
        sb.append(map.get(ch));
    }

    return Integer.parseInt(sb.toString());
}
```

"01557"





```java
for(int i = 0 ; i < 26 ; i++){
    char ch = (char)('a'+i);
    Integer res = charIntMap.get(ch);
    if(res != null){
        System.out.print(ch+"-"+res+" ");
    }
}
System.out.println();
```



Input

```
3 0 6 5 0 8 4 0 0
5 2 0 0 0 0 0 0 0
0 8 7 0 0 0 0 3 1
0 0 3 0 1 0 0 8 0
9 0 0 8 6 3 0 0 5
0 5 0 0 9 0 6 0 0
1 3 0 0 0 0 2 5 0
0 0 0 0 0 0 0 7 4
0 0 5 2 0 6 3 0 0
```

0 → Empty block

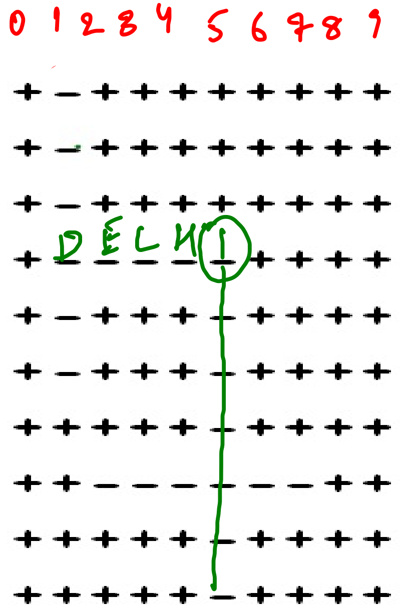
[1 – 9]

✓

Output

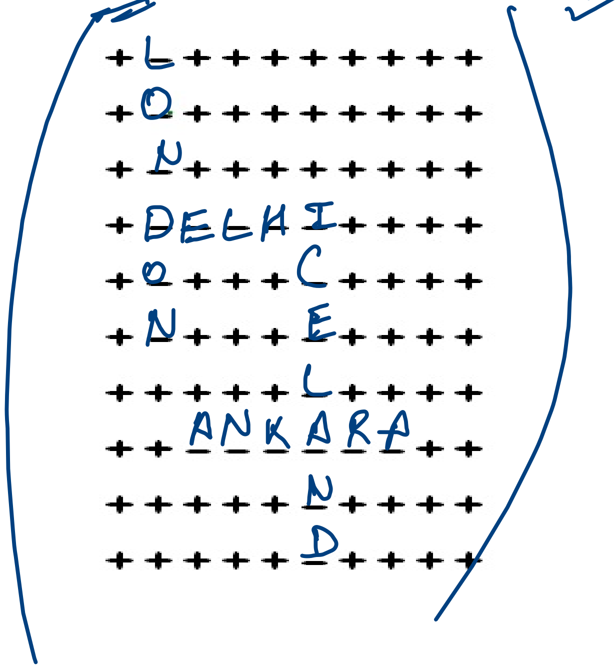
```
3 1 6 5 7 8 4 9 2
5 2 9 1 3 4 7 6 8
4 8 7 6 2 9 5 3 1
2 6 3 4 1 5 9 8 7
9 7 4 8 6 3 1 2 5
8 5 1 7 9 2 6 4 3
1 3 8 9 4 7 2 5 6
6 9 2 3 5 1 8 7 4
7 4 5 2 8 6 3 1 9
```

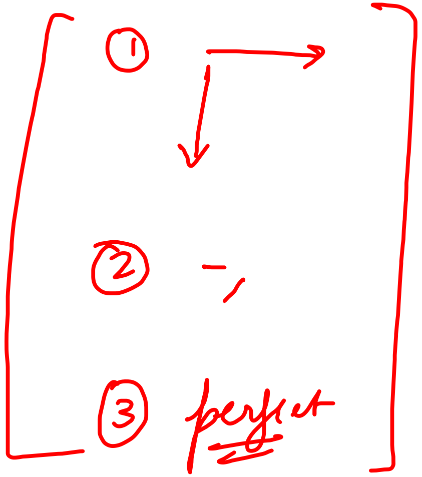
Inp



-[Word]

Output





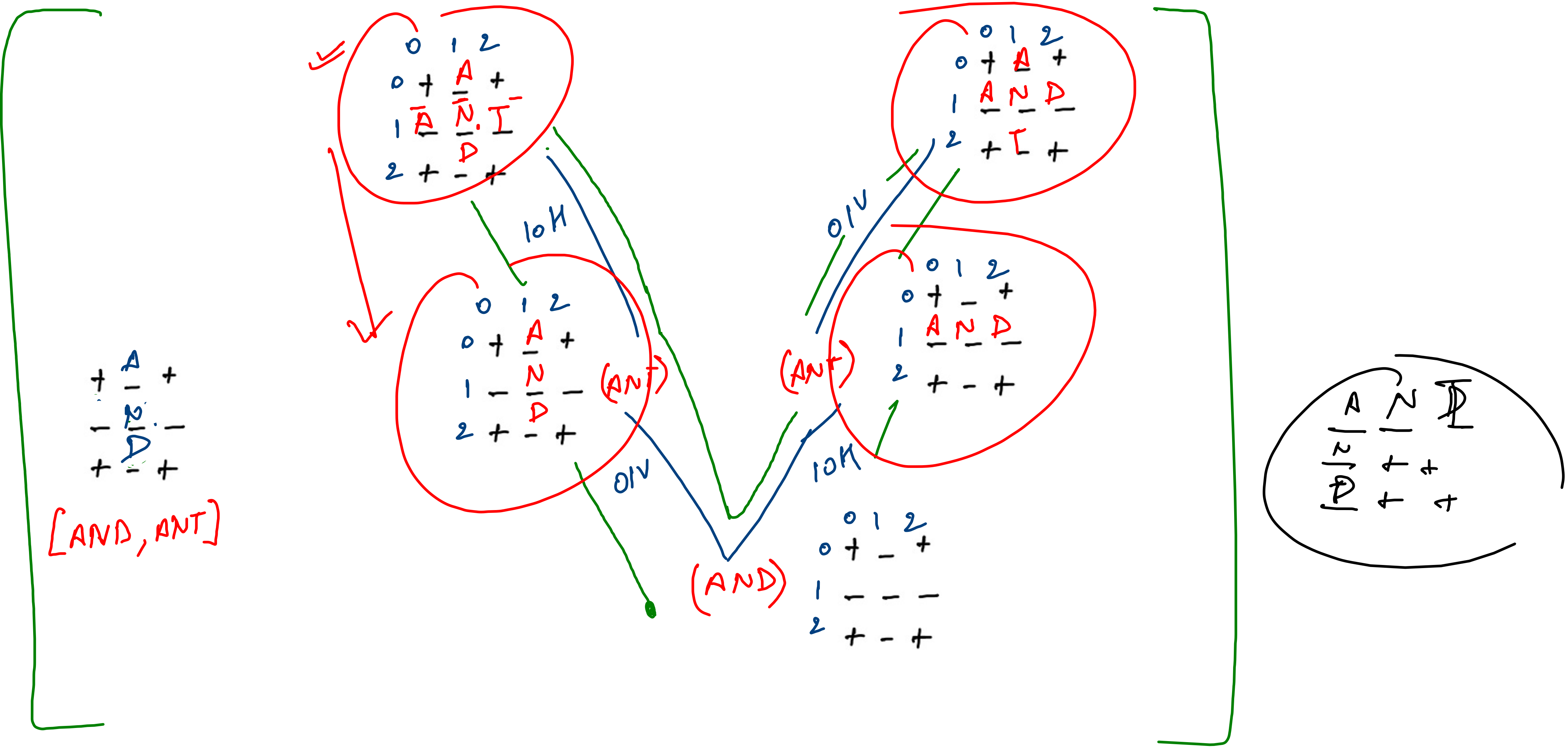
[DELHI, ICELAND, ANKARA, LONDON]

Delhi → 5

ICELAND → 7

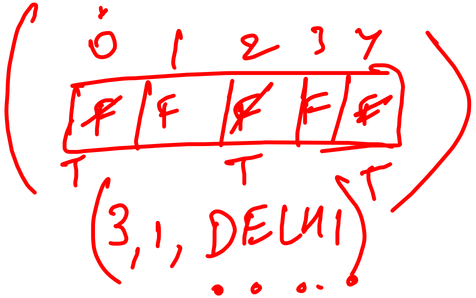
ANKARA → 6

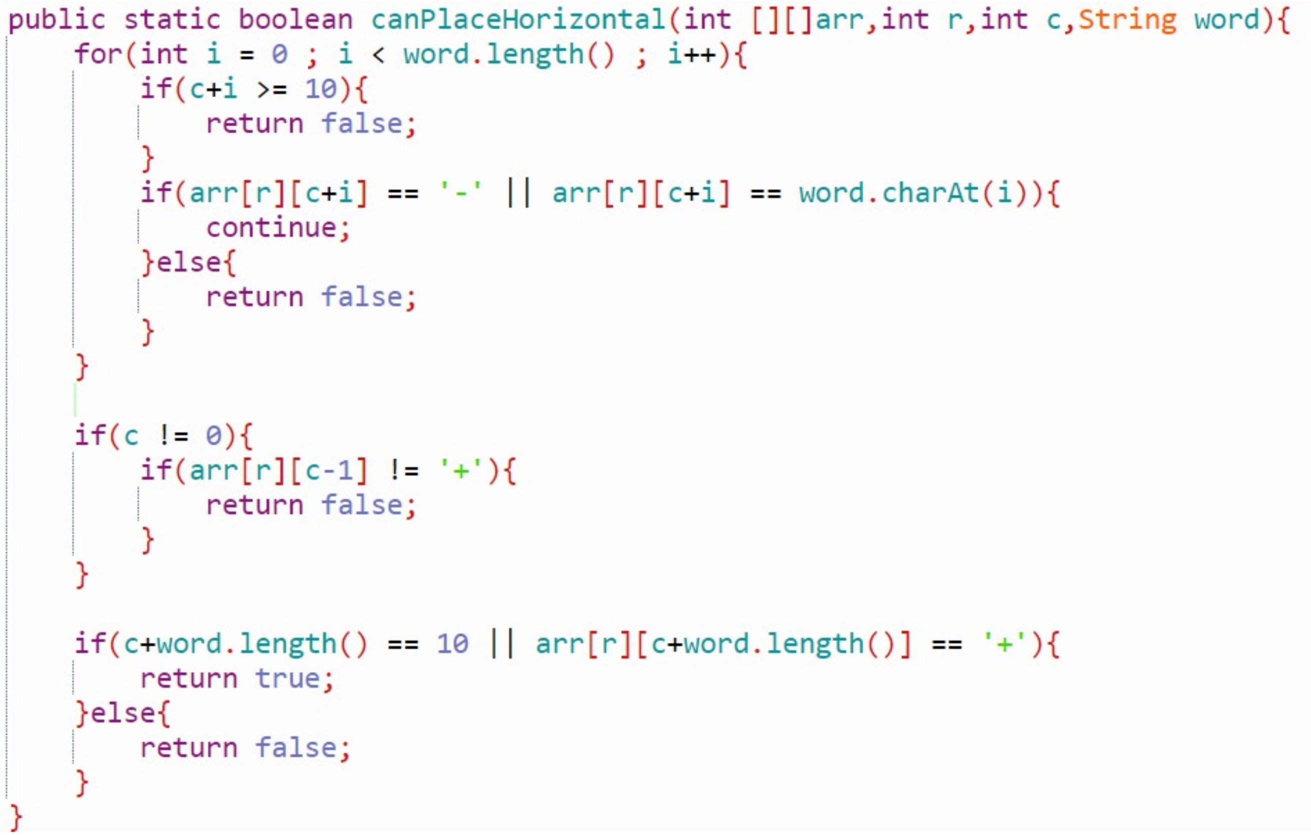
LONDON → 6


[AND, ANT]
0 1 2
10H
01V
(ANT)
(AND)
A N D
A N T

0 1 2 3 4
DELHI

```
  0 1 2 3 4 5 6 7 8 9
0 + - + + + + + + + +
1 + - + + + + + + + +
2 + - + + + + + + + +
3 + - - - - - + + + +
4 + - + + + - + + + +
5 + - + + + - + + + +
6 + + + + + - + + + +
7 + + - - - - - - + +
8 + + + + + - + + + +
9 + + + + + - + + + +
```





```java
public static boolean canPlaceHorizontal(int [][]arr,int r,int c,String word){
    for(int i = 0 ; i < word.length() ; i++){
        if(c+i >= 10){
            return false;
        }
        if(arr[r][c+i] == '-' || arr[r][c+i] == word.charAt(i)){
            continue;
        }else{
            return false;
        }
    }

    if(c != 0){
        if(arr[r][c-1] != '+'){
            return false;
        }
    }

    if(c+word.length() == 10 || arr[r][c+word.length()] == '+'){
        return true;
    }else{
        return false;
    }
}
```

9x9

	0	1	2	3	4	5	6	7	8
0	3	0	6	5	0	8	4	0	0
1	5	2	0	0	0	0	0	0	0
2	0	8	7	0	0	0	0	3	1
3	0	0	3	0	1	0	0	8	0
4	9	0	0	8	6	3	0	0	5
5	0	5	0	0	9	0	6	0	0
6	1	3	0	0	0	0	2	5	0
7	0	0	0	0	0	0	0	7	4
8	0	0	5	2	0	6	3	0	0
9									

7

$r \Rightarrow 4 \rightarrow \left(\overset{1}{r/3}\right) \times 3 \Rightarrow 3$

$c \Rightarrow 7 \rightarrow \left(\underset{2}{c/3}\right) \times 3 \Rightarrow 6$

	0	1	2	3	4	5	6	7	8
0	3	~~0~~ 1	6	5	~~0~~ 7	8	0	0	0
1	5	2	0	0	0	0	0	0	0
2	0	8	7	0	0	0	0	3	1
3	0	0	3	0	1	0	0	8	0
4	9	0	0	8	6	3	0	0	5
5	0	5	0	0	9	0	6	0	0
6	1	3	0	0	0	0	0	5	0
7	0	0	0	0	0	0	0	7	4
8	0	0	5	2	0	6	3	0	0

9 ✓

0/9 ⇒ 0 ✓

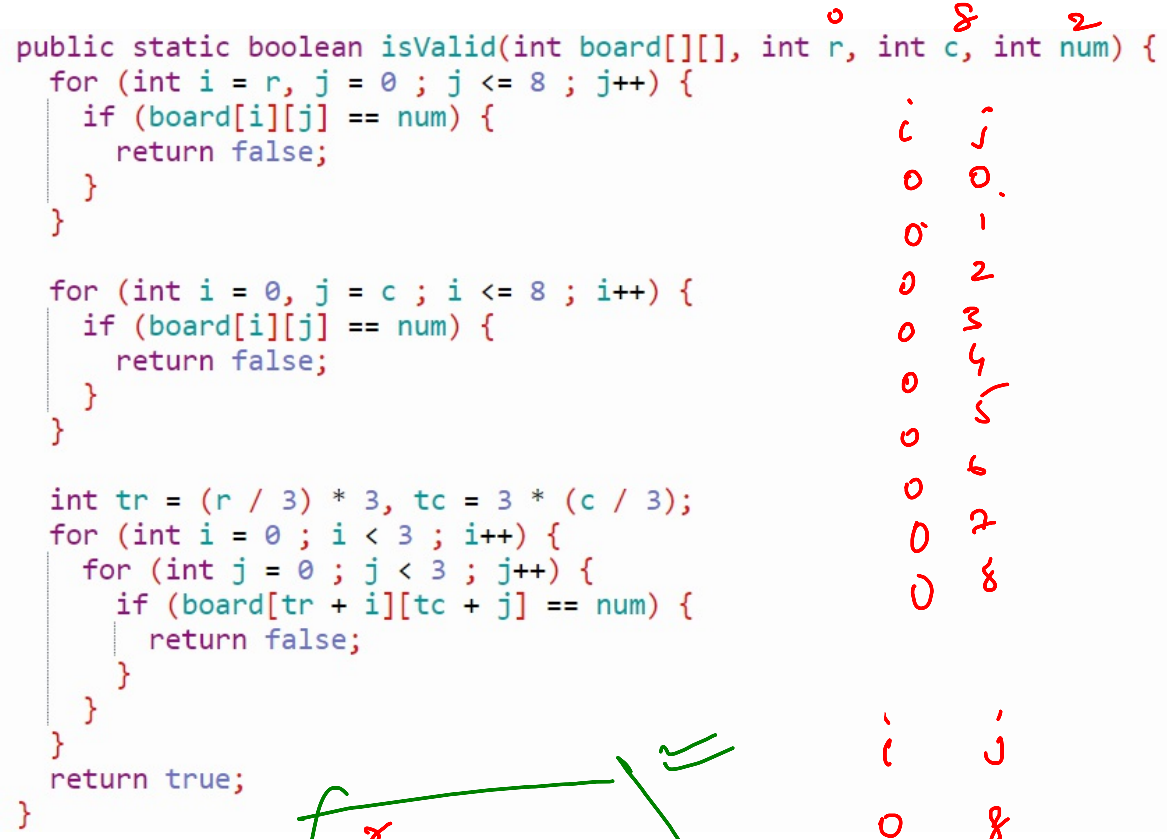
8%9 ⇒ 8 ✓

(0, 8, 2)

7/3 = 2

r	c
8	9
7,	7
↓	↓
6	6

i	j
0	0 1 2
0	6 7 8
1	0 1 2
1	6 7 8
2	0 1 2
2	6 7 8

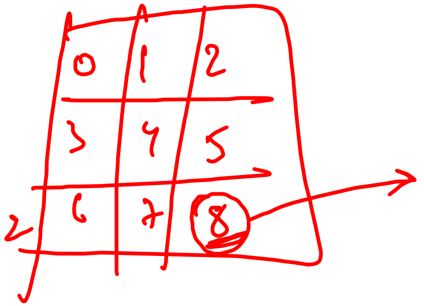


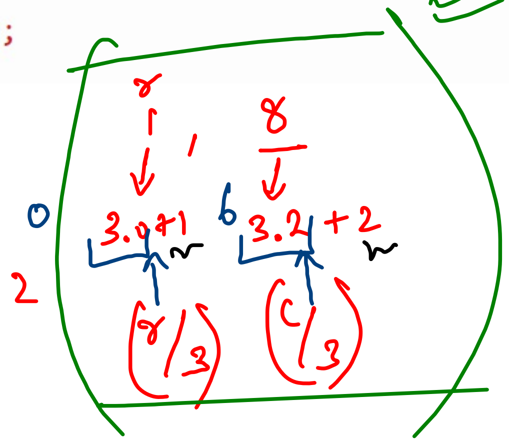
```
public static boolean isValid(int board[][], int r, int c, int num) {
  for (int i = r, j = 0 ; j <= 8 ; j++) {
    if (board[i][j] == num) {
      return false;
    }
  }

  for (int i = 0, j = c ; i <= 8 ; i++) {
    if (board[i][j] == num) {
      return false;
    }
  }

  int tr = (r / 3) * 3, tc = 3 * (c / 3);
  for (int i = 0 ; i < 3 ; i++) {
    for (int j = 0 ; j < 3 ; j++) {
      if (board[tr + i][tc + j] == num) {
        return false;
      }
    }
  }
  return true;
}
```

0 8 2

i	j
0	0
0	1
0	2
0	3
0	4
0	5
0	6
0	7
0	8

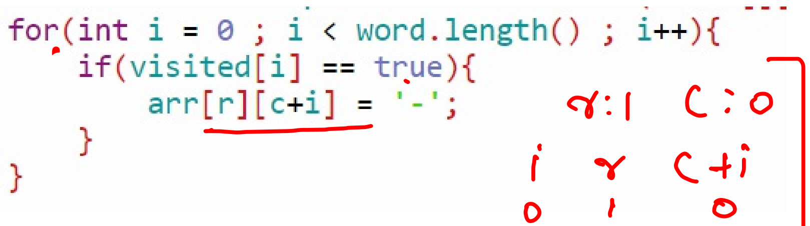
2





i	j
0	8
1	8
2	8
3	8
4	8
5	8
6	8
7	8

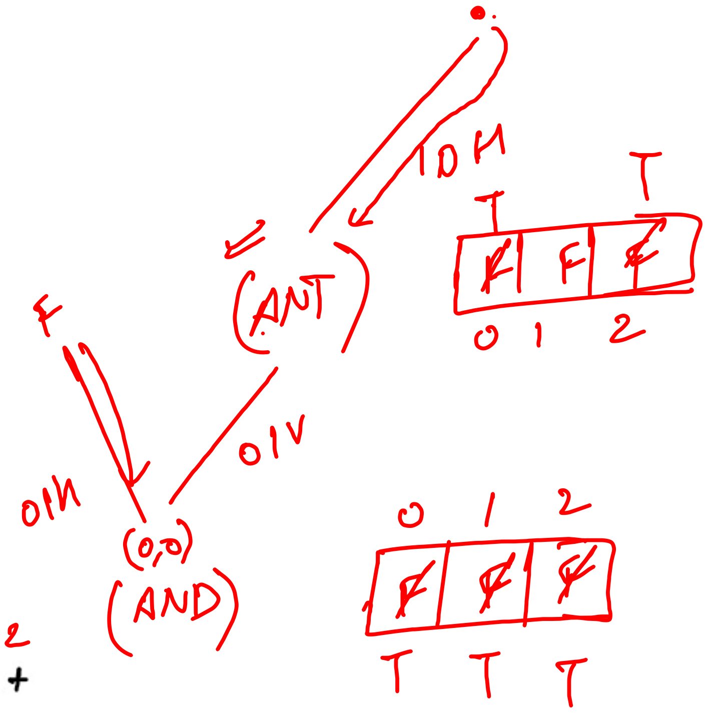
+ A +
A N T
+ D +



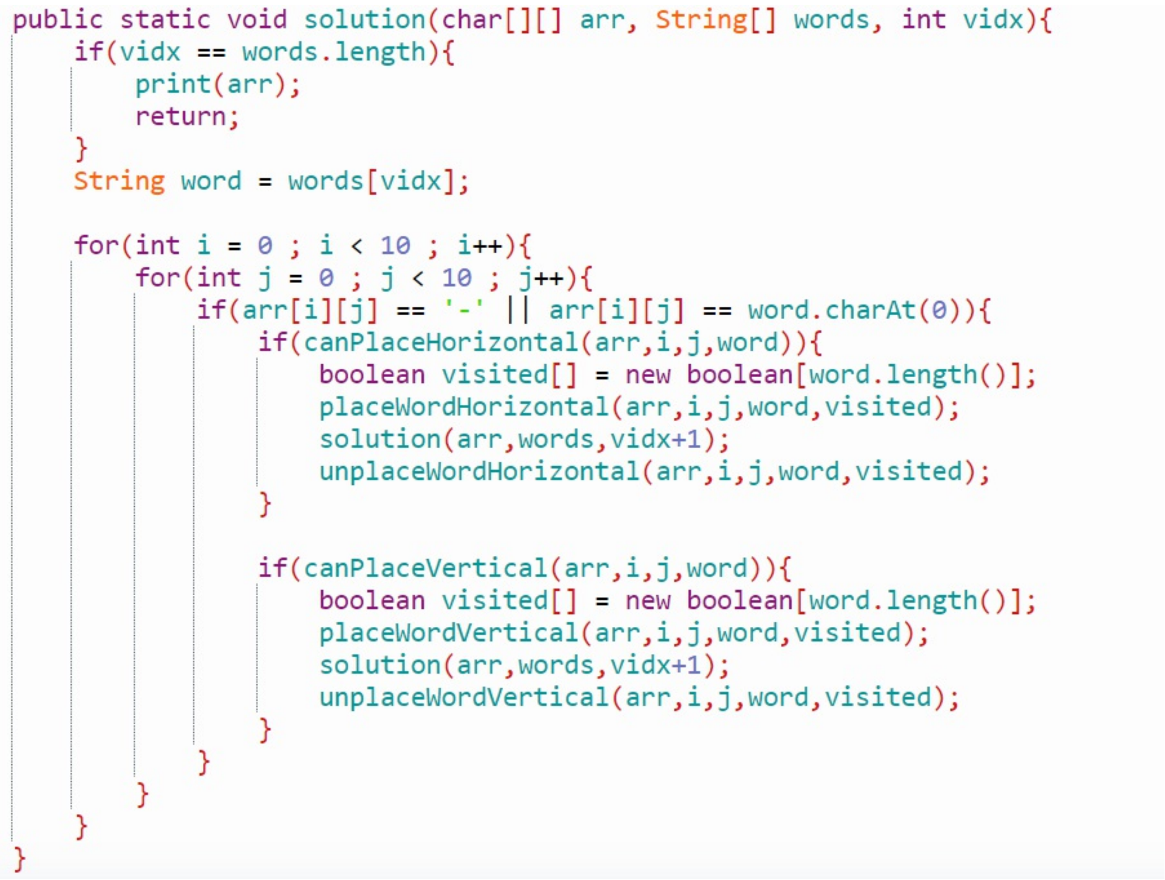
```
for(int i = 0 ; i < word.length() ; i++){
    if(visited[i] == true){
        arr[r][c+i] = '-';
    }
}
```

r:1 C:0

i	r	c+i
0	1	0
1	1	1
2	1	2



	0	1	2
0	+	A	+
1	–	N	–
2	+	D	+



```
public static void solution(char[][] arr, String[] words, int vidx){
    if(vidx == words.length){
        print(arr);
        return;
    }
    String word = words[vidx];

    for(int i = 0 ; i < 10 ; i++){
        for(int j = 0 ; j < 10 ; j++){
            if(arr[i][j] == '-' || arr[i][j] == word.charAt(0)){
                if(canPlaceHorizontal(arr,i,j,word)){
                    boolean visited[] = new boolean[word.length()];
                    placeWordHorizontal(arr,i,j,word,visited);
                    solution(arr,words,vidx+1);
                    unplaceWordHorizontal(arr,i,j,word,visited);
                }

                if(canPlaceVertical(arr,i,j,word)){
                    boolean visited[] = new boolean[word.length()];
                    placeWordVertical(arr,i,j,word,visited);
                    solution(arr,words,vidx+1);
                    unplaceWordVertical(arr,i,j,word,visited);
                }
            }
        }
    }
}
```

+ L _ _ D _ _ +

{LONDON, string}