Problem 0: Printing Number in Order Of 1 2 3 4 5 … And Order Of …5 4 3 2 1

Solution:

1 2 3 4 5…

public static void printing (int n) {  
 if(n==0) {  
  
 return;  
 }  
 *printing*(n-1);  
 System.*out*.println(n);  
  
}

…5 4 3 2 1

public static void printing (int n) {  
 if(n==0) {  
  
 return;  
 }

System.out.println(n);  
 *printing*(n-1);  
   
  
}

Problem 1: factorial of a number.

Solution:

n! = 5! = 120

private static int factorial (int n) {  
 if (n < 2 && n >= 0) {  
 return 1;  
 }  
 return n\**factorial*(n-1);  
}

Problem 2: Pow (a, b) -- > a ^ b.

Solution:

final static int *mod* = 1000000007;

public static long pow (int num, int power) {  
 // you can use mod.  
 if (power == 0) {  
 return 1;  
 }  
 long result = *pow* (num, power / 2);  
 result = (result \* result) % *mod*;  
 if (power % 2 == 1) {  
 result = (result \* num) % *mod*;  
 }  
 return result;  
}

Problem 3: LCM And HCF or GCD.

Solution:

GCD

static int gcdCal (int a, int b) {  
 if (b == 0) {  
 return a;  
 } else {  
 return *gcdCal* (b, a % b);  
 }  
}

LCM: (a \* b) / GCD

Problem 4: Fiboncci No of Nth Index:

Solution:

public static int fibonacci (int n) {  
 if (n < 2) {  
 return n;  
 }  
 return *fibonacci*(n-1) +*fibonacci*(n-2);  
}

Problem 5: Sum of two number.

Solution:

public static int sumNum (int a, int b) {   
 if (b==0) {  
 return a;  
 }  
 a++;  
 return *sumNum* (a, b-1);  
}

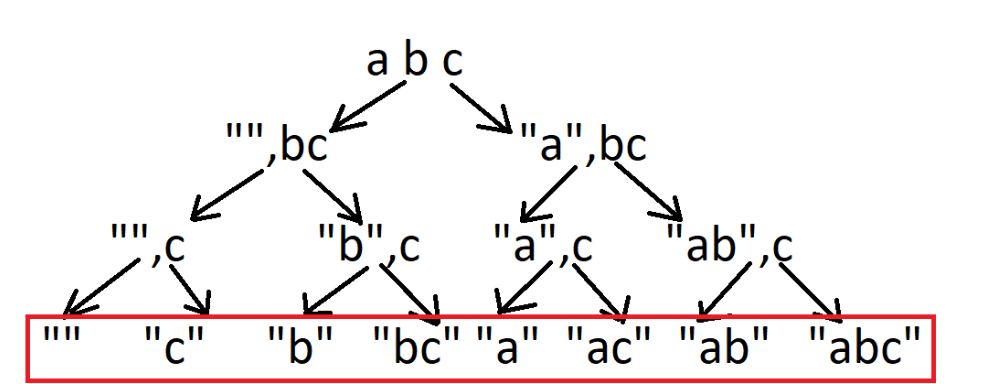
Problem 6: Subsequences Of a String.

Solution:

str = “abc,” ans = “;”



Tree:

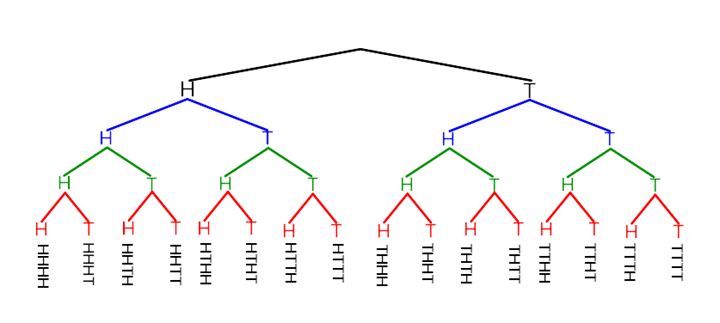


Problem 7: print all possible value getting when a coin tossed for n time.

Solution:

n = 3;

Tree:

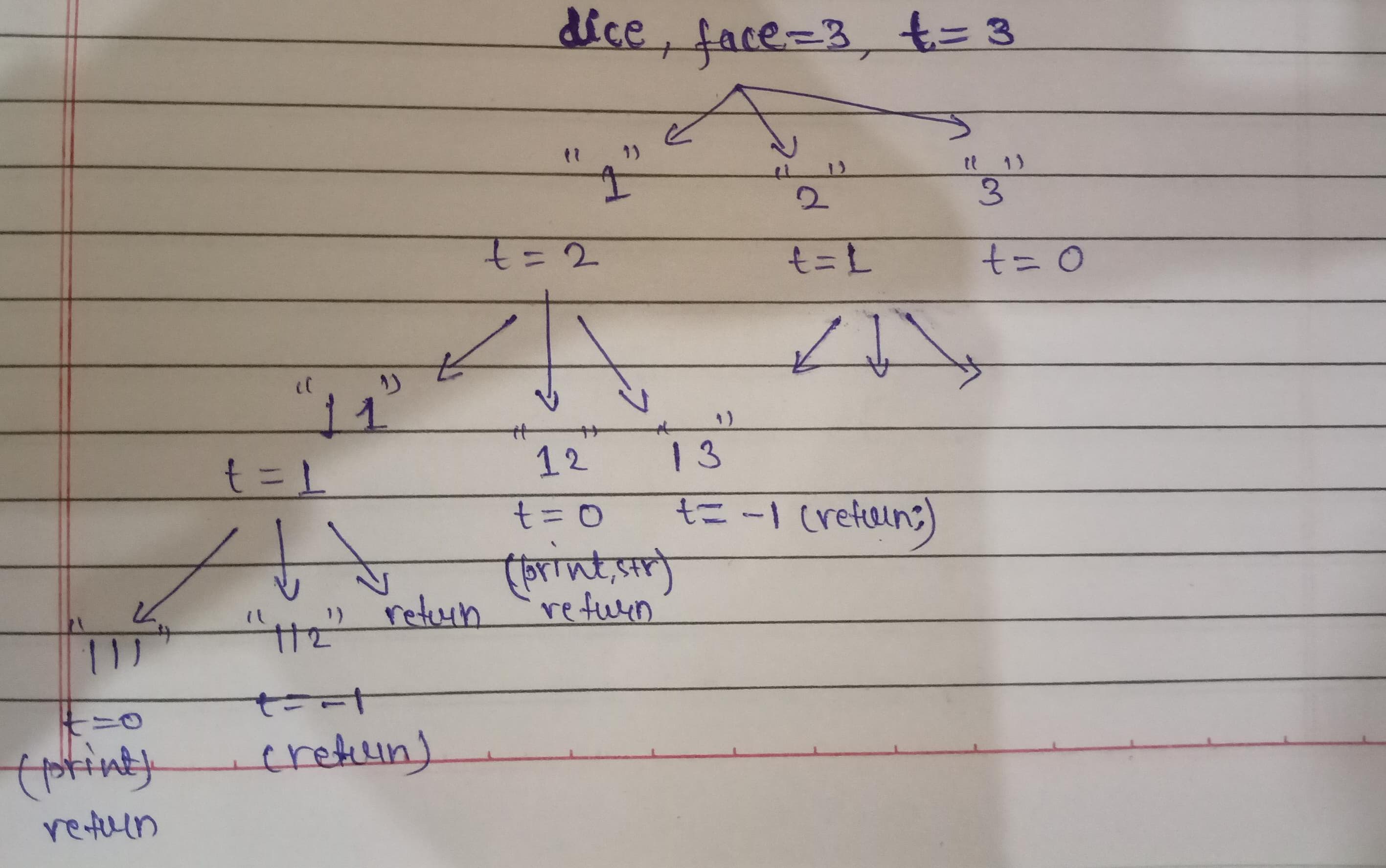


Problem 8: Dice roll for n time to get target.

Solution:

}

Tree:



Problem 9: Sum Of Digits of a Number.

Solution:

public static int sumOfNumber (int num) {  
 if (num == 0)  
 return 0;  
 return num % 10 + *sumOfNumber* (num / 10);  
}

Problem 10: code for negative even alternate sum: - 6 + 5 - 4 + 3 - 2 + 1

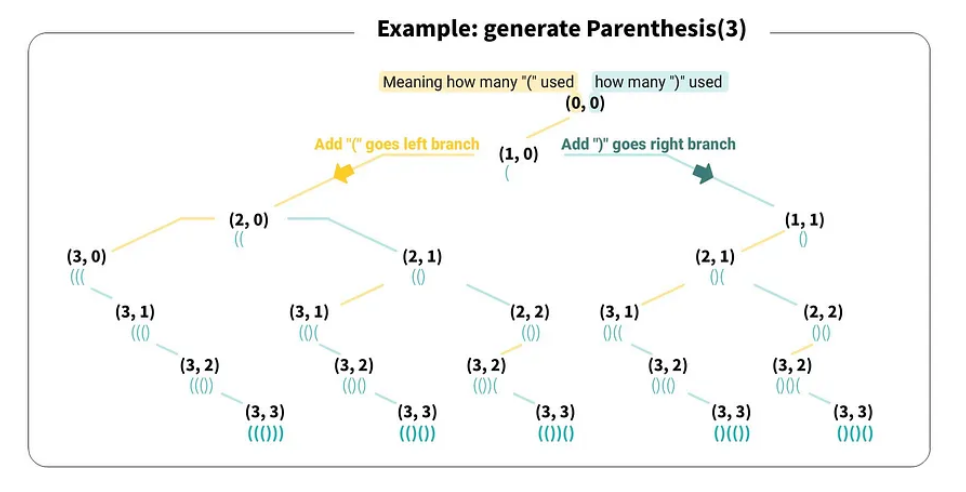
Solution:



Problem 11: [**Generate Parentheses**](https://leetcode.com/problems/generate-parentheses/) with possible permutation.

Solution:

Tree:



Problem 12: [**Unique Paths**](https://leetcode.com/problems/unique-paths/)

Solution:

M: no of row

N: no of col



Problem 13. [**Lexicographical Numbers**](https://leetcode.com/problems/lexicographical-numbers/)

Solution:

Problem 14: [**Alternating Digit Sum**](https://leetcode.com/problems/alternating-digit-sum/)

**Input:** n = 521

**Output:** 4

**Explanation:** (+5) + (-2) + (+1) = 4.

**Example 2:**

**Input:** n = 886996

**Output:** 0

**Explanation:** (+8) + (-8) + (+6) + (-9) + (+9) + (-6) = 0.

Solution:



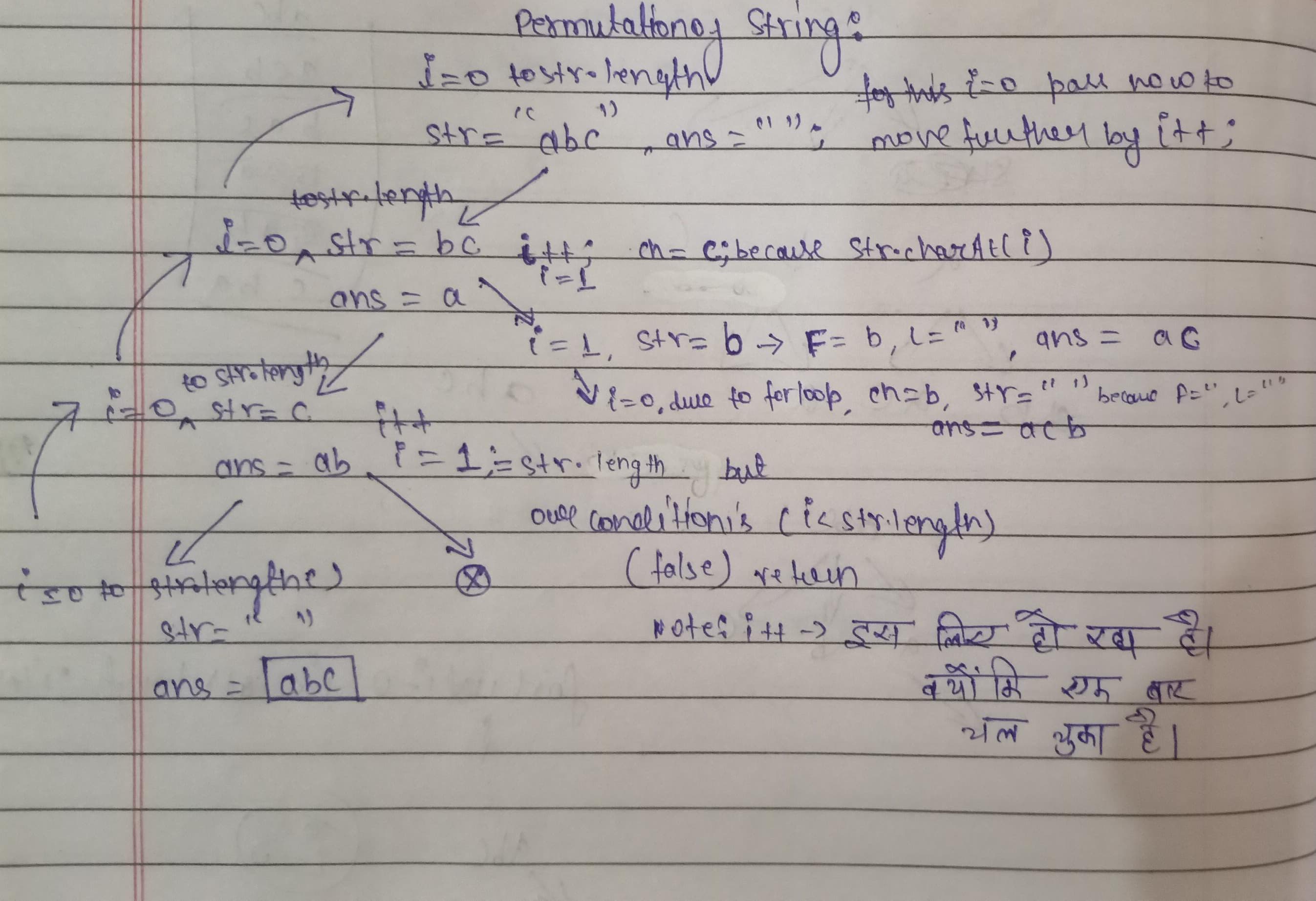
Problem 15: print all permutation of a string.

Solution:

Str = “abc”;

ans = “” ;

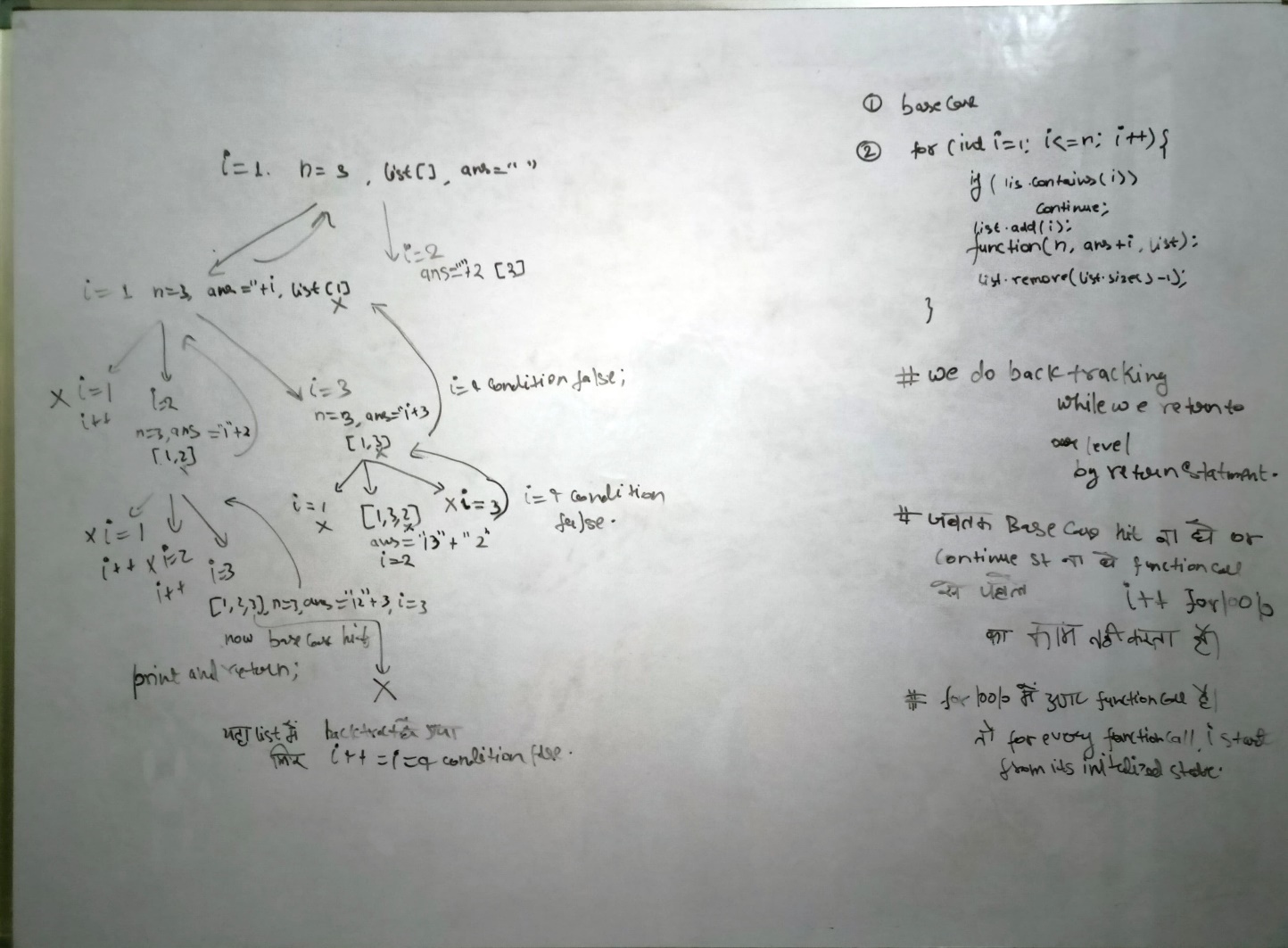
Tree:(only recursion)



for an array input:(Backtracking)



for a number input:



static void permutationNum(int n, String str, List<Integer> list) {  
 if (str.length() == 3) {  
 System.*out*.print(str + " ");  
 return;  
 }  
 for (int i = 1; i <= n; i++) {  
 if (list.contains(i)) {  
 continue;  
 }  
 list.add(i);  
 *permutationNum*(n, str + i, list);  
 list.remove(list.size()-1);  
 }  
}

Problem 19: n = 3 and k = 2 print combination of 2 element form 1 to 3.

Solution:

public static void combination1(List<List<Integer>> ls, int n, int k, List<Integer> list,int start){  
 if(list.size() == k){  
 if(!ls.contains(list))  
 ls.add(new ArrayList<>(list));  
 return;  
 }  
 for(int i = start; i <= n ; i++){  
 list.add(i);  
 *combination1*(ls,n,k,list,i+1);  
 list.remove(list.size()-1);  
 }  
}

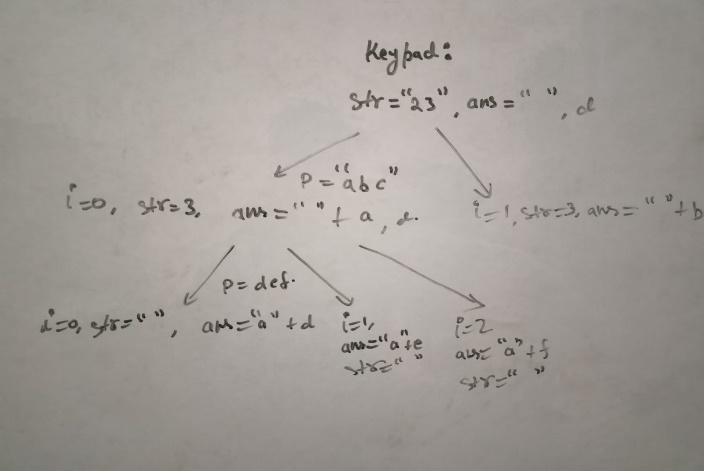
for an array:

public static void combination1(List<List<Integer>> ls, int arr[],List<Integer> list,int k, int start){  
 if(list.size() == k){  
 if(!ls.contains(list))  
 ls.add(new ArrayList<>(list));  
 return;  
 }  
 for(int i = start; i < arr.length ; i++){  
 list.add(arr[i]);  
 *combination1*(ls,arr,list,k,i+1);  
 list.remove(list.size()-1);  
 }  
}

Problem 16: [**Letter Combinations of a Phone Number**](https://leetcode.com/problems/letter-combinations-of-a-phone-number/)

Solution:

String [] d={"","abc","def","ghi","jkl","mno","pqrs","tuv","wx","yz"};  
 String s="12";  
 String ans="";  
*keypad* (s, ans, d);

public static void keypad (String s, String ans, String[] d) {  
 if (s.length() == 0) {  
 System.*out*.print(ans + " ");  
 return;  
 }  
 char ch = s.charAt(0);  
 String p = d[ch - '0'];  
 for (int i = 0; i < p.length(); i++) {  
 *keypad*(s.substring(1), ans + p.charAt(i), d);  
 }  
}

Problem 17: seating arrangement

Solution:

*seatting* (2, new boolean [] {false, false, false, false},"");

*seatting1* (2, new boolean [] {false, false, false, false},0,"",0);

// permutation:



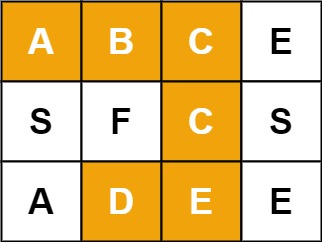
Problem 18: coin change different patterns

Solution:



Problem 20: [**Word Search**](https://leetcode.com/problems/word-search/)

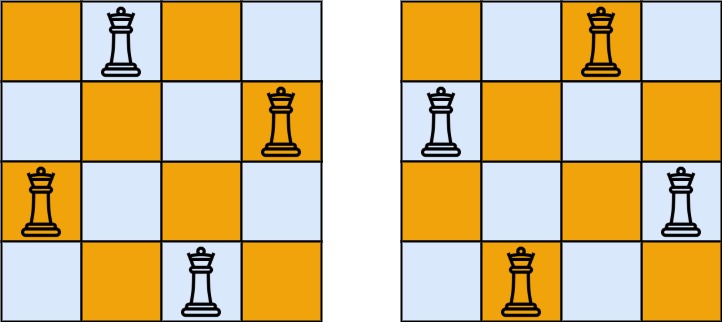
Solution:





Problem 21: [**N-Queens**](https://leetcode.com/problems/n-queens/)

Solution:



**Input:** n = 4

**Output:** [[".Q..","...Q","Q...,” “. Q."], [“. Q.","Q...","...Q",".Q.."]]


Problem 22: [**Palindrome Partitioning**](https://leetcode.com/problems/palindrome-partitioning/)

Solution:

**Input:** s = "aab"

**Output:** [["a", "a", "b"], ["aa", "b"]]



Problem 23: **Rat in a Maze Problem - I**

Solution:

**Input**:

5 4

OXOO

OOOX

OOXO

XOOO

XXOO

**Output:**

1 0 0 0

1 1 0 0

0 1 0 0

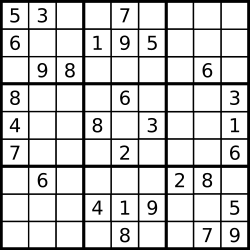
0 1 1 1

0 0 0 1

Problem 24: [**Sudoku Solver**](https://leetcode.com/problems/sudoku-solver/)

Solution:



Problem 25: Take an input N, a number. Take N more inputs and store that in an array. Take an input target, a number

a. Write a recursive function which prints subsets of the array which sum to target.

b. Write a recursive function which counts the number of subsets of the array which sum to target. Print the value returned.

Solution:



Problem 26: Take as input **N**, a number. Print odd numbers in decreasing sequence (up until 0) and even numbers in increasing sequence (up until N) using Recursion

Solution:

Input: 5

Output: 5 3 1 2 4

Problem 27:

Write a recursive function which:  
a. Returns the count of different ways the player can travel across the board. Print the value returned.  
b. Prints moves for all valid paths across the board.

Input: 3

3

Output: VVHH VHVH VHHV VHD VDH HVVH HVHV HVD HHVV HDV DVH DHV DD

13

Solution:



Problem 28:

Take as input N, a number. Write a recursive function to find Nth triangle where 1st triangle is 1, 2nd triangle is 1 + 2 = 3, 3rd triangle is 1 + 2 + 3 = 6, so on and so forth. Print the value returned.

**Input Format**

Integer **N** is the single line of input.

**Constraints**

1 <= N <= 100

**Output Format**

Print the output as a single integer which is the **nth** triangle.

**Sample Input**

3

**Sample Output**

6

Solution:

private static int triangle(int n,int sum){

if(n == 0){

return sum;

}

sum = sum + n;

return triangle(n-1,sum);

}

Problem 29:

Take an input str, a string. A “twin” is defined as two instances of a char separated by a char. E.g. "AxA" the A's make a “twin”. “twins” can overlap, so "AxAxA" contains 3 “twins” - 2 for A and 1 for x. Write a function which recursively counts number of “twins” in a string. Print the value returned.

**Input Format**

Enter the string

**Constraints**

None

**Output Format**

Display the number of twins

**Sample Input**

AXAXA

**Sample Output**

3

Solution:



**Input Format**

Enter the number of rows N and columns M

**Constraints**

None

**Output Format**

Display the total number of paths and display all the possible paths in a space separated manner

**Sample Input**

3

3

**Sample Output**

VVHH VHVH VHHV HVVH HVHV HHVV

6

Solution:



##### Problem 31: Boardpath (Count , Print)

Take as input N, a number. N is the size of a snakes and ladder board (without any snakes and ladders). Take as input M, a number. M is the number of faces of the dice.

a. Write a recursive function which returns the count of different ways the board can be traveled using the dice. Print the value returned.

b. Write a recursive function which prints dice-values for all valid paths across the board (void is the return type for function).

**Input Format**

Enter a number N (size of the board) and number M(number of the faces of a dice)

**Constraints**

M<=100  
N<=100  
M^N <=10^9

**Output Format**

Display the number of paths and print all the paths in a space separated manner

**Sample Input**

3

3

**Sample Output**

111 12 21 3

4

Solution:



##### Problem 32: Dictionary Order(Small)

Solution:



##### Problem 33: Replace all 0's with 5

Given an integer N, now you have to convert all zeros of N to 5.

**Input Format**

The first Line takes input integer N, denoting the number.

**Constraints**

1<=N<=10000

**Output Format**

Print the number after replacing all 0's with 5.

**Sample Input**

103

**Sample Output**

153

Solution:



##### Problem 34: Dictionary Order (Larger)

Take as input str, a string. Write a recursive function which prints all the words possible by rearranging the characters of this string which are in dictionary order larger than the given string.  
The output strings must be lexicographically sorted.

**Input Format**

Single line input containing a string

**Constraints**

Length of string <= 10

All characters are ***unique***

**Output Format**

Display all the words which are in dictionary order larger than the string entered in a new line each. The output strings must be sorted.

**Sample Input**

cab

**Sample Output**

cba

Solution:



##### Problem 35: Last Index?

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input M, a number. Write a recursive function which returns the last index at which M is found in the array and -1 if M is not found anywhere. Print the value returned.

**Input Format**

Enter a number N and add N more numbers to an array, then enter number M to be searched

**Constraints**

None

**Output Format**

Display the last index at which the number M is found

**Sample Input**

5

3

2

1

2

3

2

**Sample Output**

3

Solution:

v 

##### Problem 36: Recursion-Lexicographical Order

Take as input N, a number. Write a recursive function which prints counting from 0 to N in lexicographical order. In lexicographical (dictionary) order 10, 100 and 109 will be printed before 11.

**Input Format**

Enter a number N.

**Constraints**

None

**Output Format**

Display all the numbers up to N in a lexicographical order

**Sample Input**

10

**Sample Output**

0 1 10 2 3 4 5 6 7 8 9

Solution:



##### Problem 37: First Index

Take as input N, the size of array. Take N more inputs and store that in an array. Take as input M, a number. Write a recursive function which returns the first index at which M is found in the array and -1 if M is not found anywhere. Print the value returned.

**Input Format**

Enter a number N and add N more elements to an array, then enter a number M

**Constraints**

None

**Output Format**

Display the first index at which number M is found

**Sample Input**

5

3

2

1

2

2

2

**Sample Output**

1

Solution:



##### Problem 38: Generate Binary Strings

Given a string containing of ‘0’, ‘1’ and ‘?’ wildcard characters, generate all binary strings that can be formed by replacing each wildcard character by ‘0’ or ‘1’.

**Input Format**

The first line of input contains a single integer T denoting the number of test cases. Then T test cases follow. Each test case consist of two lines. The first line of each test case consists of a string S.

**Constraints**

1 ≤ T ≤ 60 1 ≤ length of string S ≤ 30

**Output Format**

Print all binary string that can be formed by replacing each wildcard character separated by space.

**Sample Input**

1

1??0?101

**Sample Output**

10000101 10001101 10100101 10101101 11000101 11001101 11100101 11101101

Solution:



Problem 39: [**200. Number of Islands**](https://leetcode.com/problems/number-of-islands/)

Solved

Medium

Topics

Companies

Given an m x n 2D binary grid grid which represents a map of '1's (land) and '0's (water), return *the number of islands*.

An **island** is surrounded by water and is formed by connecting adjacent lands horizontally or vertically. You may assume all four edges of the grid are all surrounded by water.

**Example 1:**

**Input:** grid = [

["1","1","1","1","0"],

["1","1","0","1","0"],

["1","1","0","0","0"],

["0","0","0","0","0"]

]

**Output:** 1

**Example 2:**

**Input:** grid = [

["1","1","0","0","0"],

["1","1","0","0","0"],

["0","0","1","0","0"],

["0","0","0","1","1"]

]

**Output:** 3

**Constraints:**

* m == grid. Length
* n == grid[i]. length
* 1 <= m, n <= 300
* grid[i][j] is '0' or '1'.

Solution:



Problem 40: [**463. Island Perimeter**](https://leetcode.com/problems/island-perimeter/)

You are given row x col grid representing a map where grid[i][j] = 1 represents land and grid[i][j] = 0 represents water.

Grid cells are connected **horizontally/vertically** (not diagonally). The grid is completely surrounded by water, and there is exactly one island (i.e., one or more connected land cells).

The island doesn't have "lakes", meaning the water inside isn't connected to the water around the island. One cell is a square with side length 1. The grid is rectangular, width and height don't exceed 100. Determine the perimeter of the island.

**Example 1:**



**Input:** grid = [[0,1,0,0],[1,1,1,0],[0,1,0,0],[1,1,0,0]]

**Output:** 16

**Explanation:** The perimeter is the 16 yellow stripes in the image above.

**Example 2:**

**Input:** grid = [[1]]

**Output:** 4

**Example 3:**

**Input:** grid = [[1,0]]

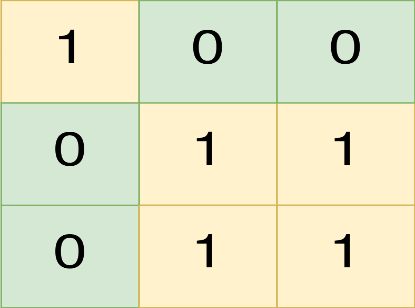
**Output:** 4

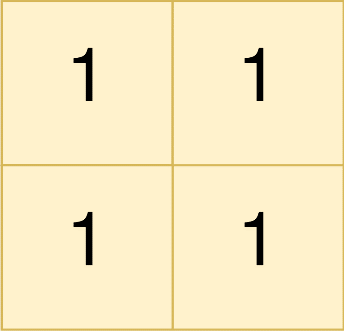
**Constraints:**

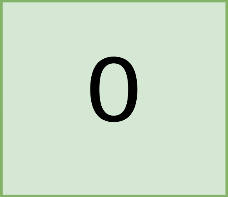
* row == grid. Length
* col == grid[i]. length
* 1 <= row, col <= 100
* grid[i][j] is 0 or 1.
* There is exactly one island in grid.

Solution:











Solution:



Problem 42. Tower Of Hanoi

Using a helper stick (peg), shift all rings from peg **A** to peg **B** using peg **C**.  
All rings are initally placed in ascending order, smallest being on top.  
No bigger ring can be placed over a smaller ring.

**Input Format**

Single line input containing a single integer N denoting the no of rings.

**Constraints**

1 <= N <= 10

**Output Format**

Print the instructions to move all the rings from peg **A** to **B** in a new line each.  
Each line should follow format : *Moving ring****i****from****A/B/C****to****A/B/C***

**Sample Input**

4

**Sample Output**

Moving ring 1 from A to C

Moving ring 2 from A to B

Moving ring 1 from C to B

Moving ring 3 from A to C

Moving ring 1 from B to A

Moving ring 2 from B to C

Moving ring 1 from A to C

Moving ring 4 from A to B

Moving ring 1 from C to B

Moving ring 2 from C to A

Moving ring 1 from B to A

Moving ring 3 from C to B

Moving ring 1 from A to C

Moving ring 2 from A to B

Moving ring 1 from C to B

Solution:

Q 



Problem 44: Split Array

Solution:

public class Main {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int arr[] = new int[n];

for(int i = 0; i < n; i++){

arr[i] = sc.nextInt();

}

fun(arr,0,0,0,new ArrayList<>(),new ArrayList<>());

System.out.print(count);

}

static int count = 0;

public static void fun(int arr[], int idx, int sum1,int sum2, List<Integer> list1, List<Integer> list2){

if(idx == arr.length){

if(sum1 == sum2){

count++;

print(list1,list2);

}

return;

}

list1.add(arr[idx]);

fun(arr,idx+1,sum1+arr[idx],sum2,list1,list2);

list1.remove(list1.size()-1);

list2.add(arr[idx]);

fun(arr,idx+1, sum1,sum2+arr[idx],list1,list2);

list2.remove(list2.size()-1);

}

public static void print(List<Integer> list1, List<Integer> list2){

for(int i = 0; i < list1.size(); i++){

System.out.print(list1.get(i)+" ");

}

System.out.print(" and ");

for(int i = 0; i < list2.size(); i++){

System.out.print(list2.get(i)+" ");

}

System.out.println();

}

}

Problem 45:

Solution:

Problem 46:

Solution:

Problem 47:

Solution:

Problem 48:

Solution:

Problem 49:

Solution:

Problem 50:

Solution:

Problem 51:

Solution:

Problem 52:

Solution:

Problem 52:

Solution: