**Letter Combinations of a Phone Number:**

public List<String> combinations(String digits) { LinkedList<String> ans = new LinkedList<String>();

if(digits.isEmpty()) return ans;  
String[] mapping = new String[] {"0", "1", "abc", "def", "ghi", "jkl", "mno", "pqrs", "tuv", "wxyz"}; ans.add("");  
for(int i =0; i<digits.length();i++){

int x = Character.getNumericValue(digits.charAt(i)); while(ans.peek().length()==i){

String t = ans.remove();  
for(char s : mapping[x].toCharArray())

ans.add(t+s);

}}

return ans;

}

**Container With Most Water**

**public int func(int[] height) {**

**int l = 0;**

**int r = height.length - 1;**

**int max = 0;**

**while(l < r){**

**int w=r - l;**

**int h= Math.min(height[l], height[r]);**

**int ar= h \* w;**

**max = Math.max(max, ar);**

**if(height[l] < height[r]) l++;**

**else if(height[l] > height[r]) r--;**

**else {l++;r--; }**

**}**

**return max; }**

**Rotate Image**

**public void image(int[][] arr) {**

**for(int i = 0; i<arr.length; i++){**

**for(int j = i; j<arr[0].length; j++){ int temp = 0;  
temp = arr[i][j];**

**arr[i][j] = arr[j][i]; arr[j][i] = temp; } }**

**for(int i =0 ; i<arr.length; i++){ for(int j = 0; j<arr.length/2; j++){**

**int temp = 0;temp = arr[i][j];  
 arr[i][j] = arr[i][arr.length-1-j]; arr[i][arr.length-1-j] = temp;**

**} }**

**}**

**Generate Parentheses**

**public List<String> brackets(int n) {**

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

**func(list, "", 0, 0, n);**

**return list;**

**}**

**public void func(List<String> list, String str, int op,int cl,int max){**

**if(str.length() == max\*2){list.add(str);return; }**

**if(op < max)**

**func(list, str+"(", op+1, cl, max);**

**if(cl < op)**

**func(list, str+")", op, cl+1, max);**

**}**

Permutations

**public List<List<Integer>> permute(int[] nums) {**

**if (nums == null || nums.length == 0)**

**return new ArrayList<>();**

**List<List<Integer>> finalResult = new ArrayList<>();**

**permuteRecur(nums, finalResult, new ArrayList<>(), new**

**boolean[nums.length]);**

**return finalResult;**

**}**

**private void permuteRecur(int[] nums, List<List<Integer>> finalResult,**

**List<Integer> currResult, boolean[] used) {**

**if (currResult.size() == nums.length) {**

**finalResult.add(new ArrayList<>(currResult));**

**return;**

**}**

**for (int i = 0; i < nums.length; i++) {**

**if (used[i])**

**continue;**

**currResult.add(nums[i]);**

**used[i] = true;**

**permuteRecur(nums, finalResult, currResult, used);**

**used[i] = false;**

**currResult.remove(currResult.size() - 1);**

**} }**