Backtracking Assignment 1 done by N S K K K Naga Jayanth

https://leetcode.com/problems/subsets/

```
class Solution {
    public List<List<Integer>> subsets(int[] nums) {
        List<List<Integer>> ans=new ArrayList<>();
        ArrayList<Integer> subset=new ArrayList<>();
        pwh(nums,0,subset,ans);
        return ans;
    }
    public void pwh(int []nums,int i,ArrayList<Integer> subset,List<List<Integer>> ans){
        if(i>=nums.length){
            ans.add(new ArrayList<>(subset));
            return;
        }

        subset.add(nums[i]);
        pwh(nums,i+1,subset,ans);
        subset.remove(subset.size()-1);
        pwh(nums,i+1,subset,ans);
    }
}
```

https://leetcode.com/problems/permutations/

```
class Solution {
    public List<List<Integer>> permute(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        finPermu(nums, 0, res);
        return res;
    }
    private void finPermu(int nums[], int startIndx, List<List<Integer>> res){
        int sz = nums.length;
        for(var num : nums){
```

```
            System.out.print(num+" ");
        }
        System.out.println();
        if(startIndx == sz){
            List<Integer> currPer = new ArrayList<>();
            for(var num : nums)currPer.add(num);
            res.add(currPer);
            return;
        }
        for(int leftPtr = startIndx ; leftPtr < sz; leftPtr++){
                swap(nums, startIndx, leftPtr);
                finPermu(nums, startIndx+1, res);
                swap(nums, startIndx, leftPtr);
        }
    }
    private void swap(int nums[], int indx1, int indx2){
        int tempNum = nums[indx1];
        nums[indx1] = nums[indx2];
        nums[indx2] = tempNum;
    }

}
```

https://leetcode.com/problems/permutations-ii/

```
class Solution {
    public List<List<Integer>> permuteUnique(int[] nums) {
         Set<List<Integer>> ans = new HashSet();
                    helper(nums,0,ans);
                    return new ArrayList(ans);
    }
     public void helper(int[] nums, int index, Set<List<Integer>> ans)
                {
```

```
            if(index==nums.length)
            {
                    ArrayList<Integer> list =new ArrayList<>();
                for(int i = 0 ; i<nums.length ; i++){
                   list.add(nums[i]);
                }
                ans.add(list);
                return;
            }

          for(int i = index; i<nums.length; i++)
          {
             swap(i,index,nums);
             helper(nums, index+1, ans);
             swap(i,index,nums);
          }
        }
        public static void swap(int i , int j, int[] nums){
              int t=nums[i];
               nums[i]=nums[j];
               nums[j]=t;
        }
}
```

https://leetcode.com/problems/subsets-ii/

```
class Solution {
    public List<List<Integer>> subsetsWithDup(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        Set<List<Integer>> res = new HashSet<>();
        List<Integer> ds = new ArrayList<>();
        Arrays.sort(nums); // Sort the input array
        fun(nums, 0, ds, res);
```

```
        for (List<Integer> subset : res) {
            ans.add(subset);
        }

        return ans;
    }

    private void fun(int[] nums, int index, List<Integer> ds, Set<List<Integer>> res) {
        if (index == nums.length) {
            res.add(new ArrayList<>(ds));
            return;
        }

        ds.add(nums[index]);
        fun(nums, index + 1, ds, res);
        ds.remove(ds.size() - 1);
        fun(nums, index + 1, ds, res);
    }
}
```

https://leetcode.com/problems/remove-invalid-parentheses/

```
class Solution {
    List<String>ans=new ArrayList<>();
    public List<String> removeInvalidParentheses(String s) {
        helper(s,0,0,'(',')');
        return ans;
    }
    void helper(String s,int is,int js,char op,char cl){
        int open=0,close=0;

        for(int i=is;i<s.length();i++){
```

```
            if(s.charAt(i)==op)open++;
            if(s.charAt(i)==cl)close++;

            if(close>open){
                for(int j=js;j<=i;j++){
                    if(s.charAt(j)==cl&&(j==js||s.charAt(j-1)!=cl)){
                        helper(s.substring(0,j)+s.substring(j+1,s.length()),i,j,op,cl);
                    }
                }
                return;
            }

        }
        String rev=new StringBuilder(s).reverse().toString();
        if(op=='('){
            helper(rev,0,0,')','(');
        }
        else{
            ans.add(rev);
        }
    }
}
```

https://leetcode.com/problems/letter-combinations-of-a-phone-number/

```
class Solution {
    private Map<Character, String> digitToLetters = new HashMap<>();
    private List<String> resultList = new ArrayList<>();

    public List<String> letterCombinations(String digits) {
        if (digits == null || digits.length() == 0) {
            return resultList;
        }
```

```
        digitToLetters.put('2', "abc");
        digitToLetters.put('3', "def");
        digitToLetters.put('4', "ghi");
        digitToLetters.put('5', "jkl");
        digitToLetters.put('6', "mno");
        digitToLetters.put('7', "pqrs");
        digitToLetters.put('8', "tuv");
        digitToLetters.put('9', "wxyz");

        generateCombinations(digits, 0, new StringBuilder());

        return resultList;


    }


    private void generateCombinations(String digits, int currentIndex, StringBuilder
currentCombination) {
        if (currentIndex == digits.length()) {
            resultList.add(currentCombination.toString());
            return;
        }

        char currentDigit = digits.charAt(currentIndex);
        String letterOptions = digitToLetters.get(currentDigit);

        if (letterOptions != null) {
            for (int i = 0; i < letterOptions.length(); i++) {
                char letter = letterOptions.charAt(i);
```

```
            currentCombination.append(letter);
            generateCombinations(digits, currentIndex + 1, currentCombination);
            currentCombination.deleteCharAt(currentCombination.length() - 1);
        }
    }
  }
}
```