Daily Practice Problem Date 6/11/2023

Problem 1:- Power of three

```
package ishwarchavan.com;

public class PowerOfThree  {      //class created
      public static void main(String[] args) {
            int n = 27;
            System.out.println(isPowerOfThree(n));
      }

    public static boolean isPowerOfThree(int n) {     //function created

       if(n==1){      //condition checking
       return true;
         }
       if(n%3!=0||n<=0){    //if condition true then return false
       return false;
    }

       return isPowerOfThree(n/3);
    }
}
```

Problem 2:- Addictive Number

```
package ishwarchavan.com;

public class AddictiveNum {

      public static void main(String[] args) {    //class created

            String num = "112358";    //num string variable created
            System.out.println(isAdditiveNumber(num));            //calling function
      }

    public static boolean isAdditiveNumber(String num) {    //function created
        int n = num.length();
                                                    // if there are less than 3 characters,
we cannot divide them to n1, n2, n3.
        if(n < 3){
            return false;
        }
                                                // iterate from 1 to n/2+1. Because if n is
10, then n1 can max be of 10/2 = 5 length.
        for(int i=1; i<n/2+1; i++){
            long n1 = Long.parseLong(num.substring(0, i));
            if(!String.valueOf(n1).equals(num.substring(0, i))){
                break;
            }
                                                    // j will run till n. Because, n2
might have more digits than n1.
            for(int j=i+1; j<n; j++){
                long n2 = Long.parseLong(num.substring(i, j));
                if(!String.valueOf(n2).equals(num.substring(i, j))){
                    break;
                }
                                                        // recursively match the
preceding characters.
                boolean found = false;
                        try {
                              found = backtrack(n1, n2, num.substring(j), false);
                        } catch (Exception e) {
```

```
                                e.printStackTrace();
                            }
                        if(found){
                            return true;
                        }
                    }
                }
                return false;
            }

            static boolean backtrack(long n1, long n2, String num, boolean found){  //function
created
                if(num.length() == 0 && found){
                    return true;
                }
                String n3 = String.valueOf(n1+n2);
                                                                    // n3 might have more
length than num. So, get the minimum length.
                int idx = Math.min(num.length(), n3.length());

                                                                  // if the substring is equal to
n3, then we can further proceed.
                if(num.substring(0, idx).equals(n3)){
                    return backtrack(n2, Long.parseLong(n3), num.substring(idx), true);
                }
                return false;
            }
        }
```