Question-1:

Given an input string and a dictionary of words, find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.

This is a famous Google interview question, also being asked by many other companies now a days.

Consider the following dictionary

{ i, like, sam, sung, samsung, mobile, ice,

cream, icecream, man, go, mango}

Input: ilike

Output: Yes

The string can be segmented as "i like".

Input: ilikesamsung

Output: Yes

The string can be segmented as "i like samsung"

or "i like sam sung".

```java
import java.util.ArrayList;
import java.util.Scanner;


public class WordPresent {
   public static void main(String[] args) {
       Scanner sc = new Scanner(System.in);


       int n = sc.nextInt();
       ArrayList<String> al = new ArrayList<>();
```

```
        for(int i = 0; i < n; i++){
            al.add(sc.next());
        }
        String word = sc.next();
        if(checkWord(word, al)){
            System.out.println("Yes");
        }
        else{
            System.out.println("No");
        }
    }
   static boolean checkWord(String word,
ArrayList<String> al){


        if(word.length() == 0){
           return true;
        }
        for(int i = 1; i <= word.length(); i++){
            if(al.contains(word.substring(0,i)) &&
checkWord(word.substring(i),al)){
                return true;
            }
        }
        return false;
```

```
    }
}
//OUTPUT
// 12
// i like sam sung samsung mobile ice cream icecream man go mango
// ilikesamsung
// Yes
```

Ques-2

A number can always be represented as a sum of squares of other numbers. Note that 1 is a square and we can always break a number as (1*1 + 1*1 + 1*1 + …). Given a number n, find the minimum number of squares that sum to X.

Examples :

Input: n = 100

Output: 1

Explanation:

100 can be written as 102. Note that 100 can also be written as 52 + 52 + 52 + 52, but this representation requires 4 squares.

Input: n = 6

Output: 3

```
import java.util.Scanner;


public class SquareSum {
```

```java
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        System.out.println(findMinSquareSum(n));
    }
    static int findMinSquareSum(int n){
        //Recursive
//            if(n < 4){
//                return n;
//            }
//
//        int res = n;
//        for (int i = 1; i <= n; i++) {
//            int temp = i * i;
//            if (temp > n)
//                break;
//            else
//                res = Math.min(res, 1 +
findMinSquareSum(n - temp));
//        }
//        return res;
        // Tabulation (Bottom Up DP)
        int[] dp = new int[n + 1];
        dp[0] = 0;
```

```
        for (int i = 1; i <= n; i++) {
            dp[i] = i;

            for (int j = 1; j * j <= i; j++) {
                int square = j * j;
                dp[i] = Math.min(dp[i], 1 + dp[i -
square]);
            }
        }

        return dp[n];
    }
}
//OUTPUT
// 6(input)
// 3


// 100(input)
// 1
```

Ques-3

Given a number N, the task is to check if it is divisible by 7 or not.

Note: You are not allowed to use the modulo operator, floating point arithmetic is also not allowed.

Naive approach: A simple method is repeated subtraction. Following is another interesting method.

Divisibility by 7 can be checked by a recursive method. A number of the form 10a + b is divisible by 7 if and only if a – 2b is divisible by 7. In other words, subtract twice the last digit from the number formed by the remaining digits. Continue to do this until a small number.

Example: the number 371: 37 – (2×1) = 37 – 2 = 35; 3 – (2 × 5) = 3 – 10 = -7; thus, since -7 is divisible by 7, 371 is divisible by 7.

```java
import java.util.Scanner;


public class DivisibleBy7 {

   public static void main(String[] args) {

       Scanner sc = new Scanner(System.in);

       int n = sc.nextInt();

       if(checkDivisible(n)){

           System.out.println("No. is divisible by 7");

       }

       else{

           System.out.println("Not divisible by 7");

       }

   }
```

```
   static boolean checkDivisible(int n){


       if(n < 0){

          return checkDivisible(-n);

       }

       if(n == 0 || n== 7){

           return true;

       }

       if(n < 10){

           return false;

       }

       return checkDivisible(n/10 - 2 * (n - (n/10) *
10));

   }

}

//OUTPUT

//371(input)

//No. is divisible by 7


//15(input)

//Not divisible by 7
```

Question-4

Find the n’th term in Look-and-say (Or Count and Say) Sequence. The look-and-say sequence is the sequence of the below integers:

1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, …

How is the above sequence generated?

n’th term is generated by reading (n-1)’th term.

The first term is "1"

Second term is "11", generated by reading first term as "One 1"

(There is one 1 in previous term)

Third term is "21", generated by reading second term as "Two 1"

Fourth term is "1211", generated by reading third term as "One 2 One 1"

and so on

Input: n = 3

Output: 21

Input: n = 5

Output: 111221

```
import java.util.Scanner;


public class Sequence {

   public static void main(String[] args) {

        Scanner sc = new Scanner(System.in);

        int n = sc.nextInt();


        System.out.println(findTermInSequence(n));
```

```
    }
    static String findTermInSequence(int n){

        if( n == 0 ) return "";
        String ans = "1";
        if( n == 1 )
            return ans;
        n = n-1;
        while(n-- > 0) {
            String temp = "" ;
            for(int i = 0 ; i < ans.length() ; i++) {
                Integer cnt = 1;
                while(i < ans.length()-1 &&
ans.charAt(i) == ans.charAt(i+1)) {
                    cnt++;
                    i++;
                }
                temp += cnt.toString() + ans.charAt(i);
            }
            ans = temp;
        }
        return ans;
    }
}
```

```
// OUTPUT

// 3(input)

// 21



// 5(input)

// 111221
```