Day 2 Practical Work

Mridul Bhardwaj

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".

Solution :

```
public class WordSegmentation {
    public static void main(String[] args) {
        String input1 = "idontlikesung";
        String input2 = "ilikesamsung";
        String[] dictionary = {"i", "like", "sam", "sung", "samsung", "mobile", "ice", "cream",
"icecream", "man", "go", "mango"};

        if (canSegment(input1, dictionary)) {
            System.out.println("Output for " + input1 + ": Yes");
        } else {
            System.out.println("Output for " + input1 + ": No");
        }
        if (canSegment(input2, dictionary)) {
            System.out.println("Output for " + input2 + ": Yes");
        } else {
            System.out.println("Output for " + input2 + ": No");
        }
    }

    static boolean canSegment(String input, String[] dictionary) {
        if (input.isEmpty()) {
            return true;
        }
        for (String word : dictionary) {
            if (input.startsWith(word)) {

                if (canSegment(input.substring(word.length()), dictionary)) {
                    return true;
                }
            }
        }
        return false;
    }
}
```

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

Solution :

```
import java.util.*;

import java.io.*;


public class MinSquare
```

```
{
static int getMinSquares(int n)
  {
    if (n <= 3)
      return n;
    int res = n;
    for (int x = 1; x <= n; x++)
    {
      int temp = x * x;
      if (temp > n)
        break;
      else
        res = Math.min(res, 1 +
             getMinSquares(n - temp));
    }
    return res;
  }
       public static void main(String[] args) {
         Scanner sc = new Scanner(System.in);
         System.out.println("Enter the number:");
         int num = sc.nextInt();
             System.out.println("Minimum Squares are: " + getMinSquares(num));
       }
}
```

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.

Solution :

```
import java.util.*;
import java.io.*;

public class Divisible7
{
static boolean isDivisibleBy7(int n)
   {
      if( n < 0 )
         return isDivisibleBy7( -n );
      if( n == 0 || n == 7 )
         return true;
      if( n < 10 )
         return false;
      return isDivisibleBy7( n / 10 - 2 * ( n - n / 10 * 10 ) );
   }
          public static void main(String[] args) {
             Scanner sc = new Scanner(System.in);
             System.out.println("Enter the number:");
             int num = sc.nextInt();
                    if(isDivisibleBy7(num))
         System.out.println("Divisible");
```

```
        else
            System.out.println("Not Divisible");
             }
}
```

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

Solution :

```
import java.util.Scanner;

public class LookAndSay{
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String result = lookAndSay(n);
        System.out.println("The " + n + "'th term in the Look-and-say sequence is: " + result);
    }
```

```
static String lookAndSay(int n) {
    if (n <= 0) {
        return "Invalid input";
    }
    if (n == 1) {
        return "1";
    }

    String previousTerm = "1";
    for (int i = 2; i <= n; i++) {
        previousTerm = getNextTerm(previousTerm);
    }
    return previousTerm;
}
static String getNextTerm(String s) {
    StringBuilder result = new StringBuilder();
    int count = 1;

    for (int i = 1; i < s.length(); i++) {
        if (s.charAt(i) == s.charAt(i - 1)) {
            count++;
        } else {
            result.append(count).append(s.charAt(i - 1));
```

```
                count = 1;
            }
        }
        result.append(count).append(s.charAt(s.length() - 1));
        return result.toString();
    }
}
```