WIX1002: FOP
Viva 1 : Flow Control

Group Member

25069629	YANG YU TING
25071951	LI YUCHEN
25065764	CHENYUHAN
24211805	MENGHANYUE
25064763	LIZHAOZIYU
24088371	Tahzib Farhan

Question 1: The Tok Wan's Number Charms and the Pasar Malam Challenge

Maintainer: 25069629 YANG YU TING

1) Problem

Tok Wan creates a sequence called a “number charm” using three numbers:

- Initial Value (a)
- Multiplier Seed (b)
- Charm Length (n)

You will receive q queries, and each query gives a, b, n.

For each query, you must generate a sequence of n numbers, based on the formula:

Value = a + b*2^i

2) Solution

Step 1 — Read the number of queries q

This tells us how many sequences we need to generate.

Step 2 — For each query:

Read a, b, and n.

Step 3 — Generate the sequence

Use a loop that runs from i = 0 to n−1.

At each step:

- Compute the exponential term:

  2^i

- Multiply it by b

  b*2^i

- Add to a to get the charm number.

Step 4 — Print the numbers

Each sequence is printed on a new line, numbers separated by spaces.

3) Sample Input & Output

Given:

a=1

b=4

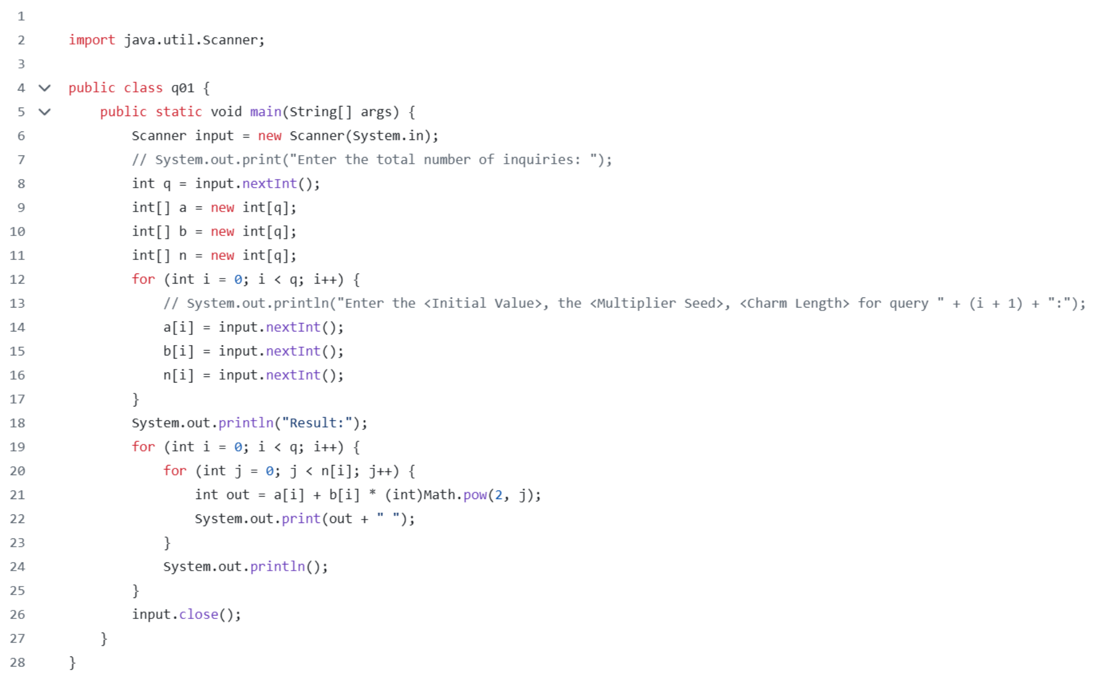
n=4

Steps:

- $i = 0 \rightarrow 1 + 4 \times 1 = 5$
- $i = 1 \rightarrow 1 + 4 \times 2 = 9$
- $i = 2 \rightarrow 1 + 4 \times 4 = 17$
- $i = 3 \rightarrow 1 + 4 \times 8 = 33$

Final sequence: 5 9 17 33

4) Source Code



```java

import java.util.Scanner;

public class q01 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // System.out.print("Enter the total number of inquiries: ");
        int q = input.nextInt();
        int[] a = new int[q];
        int[] b = new int[q];
        int[] n = new int[q];
        for (int i = 0; i < q; i++) {
            // System.out.println("Enter the <Initial Value>, the <Multiplier Seed>, <Charm Length> for query " + (i + 1) + ":");
            a[i] = input.nextInt();
            b[i] = input.nextInt();
            n[i] = input.nextInt();
        }
        System.out.println("Result:");
        for (int i = 0; i < q; i++) {
            for (int j = 0; j < n[i]; j++) {
                int out = a[i] + b[i] * (int)Math.pow(2, j);
                System.out.print(out + " ");
            }
            System.out.println();
        }
        input.close();
    }
}
```

Question 2 Report: Ah Hock's Digital Signature

Maintainer: 25064763   LIZHAOZIYU

1) Problem

Ah Hock needs to determine the digital signature of numbers based on their digits and a lucky digit. For a given number N and lucky digit L, the program must:

- Categorize each digit of N according to priority: lucky digits, zeros, even digits, odd digits
- Count the number of digits in each category
- Output the corresponding signature (LUCKY, BALANCED, ENERGETIC, or NEUTRAL) based on which category has the highest count

2) Solution

My solution involves three main steps:

1. Read input data, including the number of test cases and each test case's number N and lucky digit L
2. Convert number N to a string, iterate through each digit and count them according to priority categories
3. Compare the counts of the four categories to determine and output the corresponding digital signature

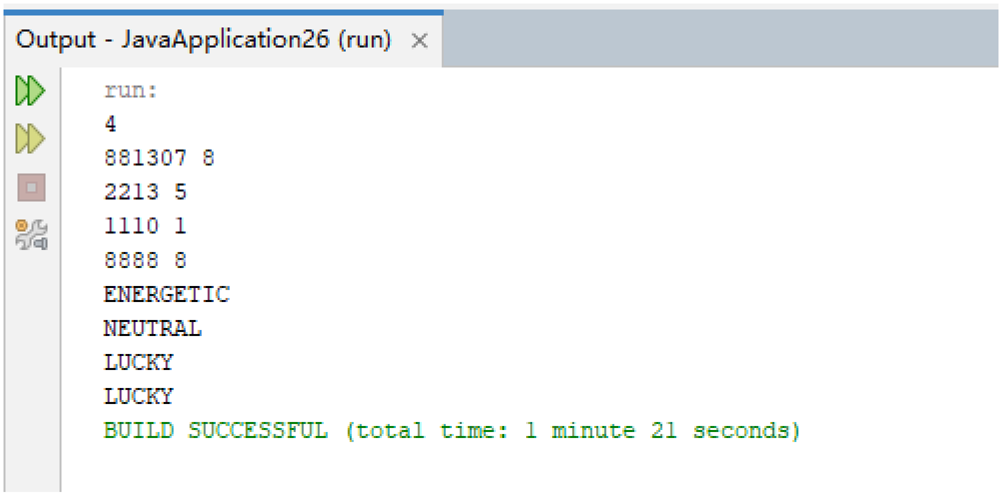
Key points:

- Use if-else if structure to ensure classification priority

- Strictly compare counts to determine signatures

- Handle special cases (e.g., when L=0, all zeros are considered lucky digits)

3) Sample Input & Output


Output - JavaApplication26 (run)

```
run:
4
881307 8
2213 5
1110 1
8888 8
ENERGETIC
NEUTRAL
LUCKY
LUCKY
BUILD SUCCESSFUL (total time: 1 minute 21 seconds)
```

4) Code

```java

import java.util.Scanner;

public class q02 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // System.out.print("Enter the total number of inquiries: ");
        int q = input.nextInt();
        int[] n = new int[q];
        int[] l = new int[q];
        for (int i = 0; i < q; i++) {
            // System.out.println("Enter the <Number> and <Lucky Digit> for query " + (i + 1));
            n[i] = input.nextInt();
            l[i] = input.nextInt();
        }
        for (int i = 0; i < q; i++) {
            int lucky = 0;
            int zero = 0;
            int odd = 0;
            int even = 0;
            while (n[i] != 0) {
                int temp = n[i] % 10;
                n[i] /= 10;
                if (temp == l[i]) lucky++;
                else if (temp == 0) zero++;
                else if (temp % 2 == 1) odd++;
                else even++;
            }
            if (lucky > zero && lucky > odd && lucky > even) System.out.println("LUCKY");
            else if (even > lucky && even > odd & even > zero) System.out.println("BALANCED");
            else if (odd > lucky && odd > zero && odd > even) System.out.println("ENERGETIC");
            else System.out.println("NEUTRAL");
        }
        input.close();
    }
}
```

Question 3 Puan Norah's Digital Kolam

Maintainer: 24211805　MENGHANYUE

1) Problem

There are two primary styles:**Style 'A' (Angled)/Style 'P' (Pyramid).**

A:print H row and Row i (where i is from 1 to H) contains the digit i repeated i times.

P:(1)print H row

(2)Each row must be padded with leading spaces to be centered. A pyramid of　H has a base width of (2×H)−1.

(3)Row i (where i is from 1 to H) will consist of:

1. A number of leading spaces.
2. Numbers ascending from 1 up to i.
3. Numbers descending from i−1 back down to 1

2) Solution

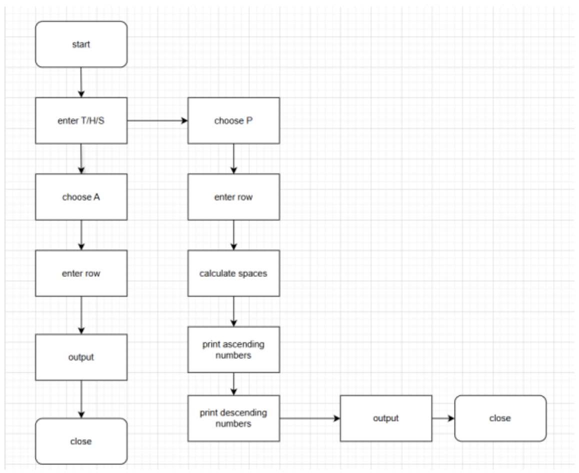
First choose style (A/P)

Second input total times

Choose A input H repeat i

Choose P int row ,spaces,number from 1 up to i and from i−1 back down to 1.

3) Sample Input & Output



4) Source code

```java
import java.util.Scanner;

public class q03 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // System.out.println("Enter the total number of inquiries: ");
        int q = input.nextInt();
        int[] height = new int[q];
        String[] style = new String[q];
        for (int i = 0; i < q; i++) {
            // System.out.println("Enter the <height> and the <style> for query " + (i + 1));
            height[i] = input.nextInt();
            style[i] = input.next();
        }

        for (int i = 0; i < q; i++) {
            if (style[i].equalsIgnoreCase("A")) {
                for (int j = 1; j <= height[i]; j++) {
                    for (int k = 1; k <= j; k++) System.out.print(j);
                    System.out.println();
                }
            } else if (style[i].equalsIgnoreCase("P")) {
                for (int j = 1; j <= height[i]; j++) {
                    for (int k = 1; k <= height[i] - j; k++) {
                        System.out.print(" ");
                    }
                    for (int k = 1; k <= j; k++) {
                        System.out.print(k);
                    }
                    for (int k = j - 1; k >= 1; k--) {
                        System.out.print(k);
                    }
                    System.out.println();
                }
            } else System.out.println("Invaild style for query " + (i + 1));
        }
        input.close();
    }
}
```

Question 4: The Tok Wan's Word Gems

Maintainer: 24088371 Tahzib Farhan

1)Problem

Tok Wan Osman believes that hidden "word gems" can be found inside any word by examining all possible substrings of a given fixed length k.

For each substring of length k, we must identify three gems:

1. First Whisper – The lexicographically smallest substring.
2. Last Echo – The lexicographically largest substring.
3. Core Value – The substring with the highest ASCII total sum.

2)Solution

Step-01:

We read the input, convert it to lowercase, and read integer k, which is the required substring length.

Step-02:

We generate all substrings of length k.

If the word length is n, the total substrings is n-k +1.

Step-03:

We track the three special gems.

- We maintain firstWhisper = lexicographically smallest substring.
- We maintain lastEcho = lexicographically largest substring.
- We maintain coreValueSubstring = substring with the highest ASCII sum.
- We maintain maxAsciiSum = highest sum found so far.

We compare each substring-

A. Finding First Whisper: -

   Compare substring using: if (current.compareTo(firstWhisper) < 0).

B. Finding Last Echo: -

   if (current.compareTo(lastEcho) > 0).

C. Finding Core Value: -

   Compute ASCII sum character by character: sum = c1 + c2 + c3 + ...

   If:

   sum > maxAsciiSum

   then, update both. If equal, we keep the earlier substring as stated.

<u>Step-04:</u>

We print the output.

The output is printed in the order-

1) First Whisper
2) Core Value
3) Last Echo

3)Sample Input & Output

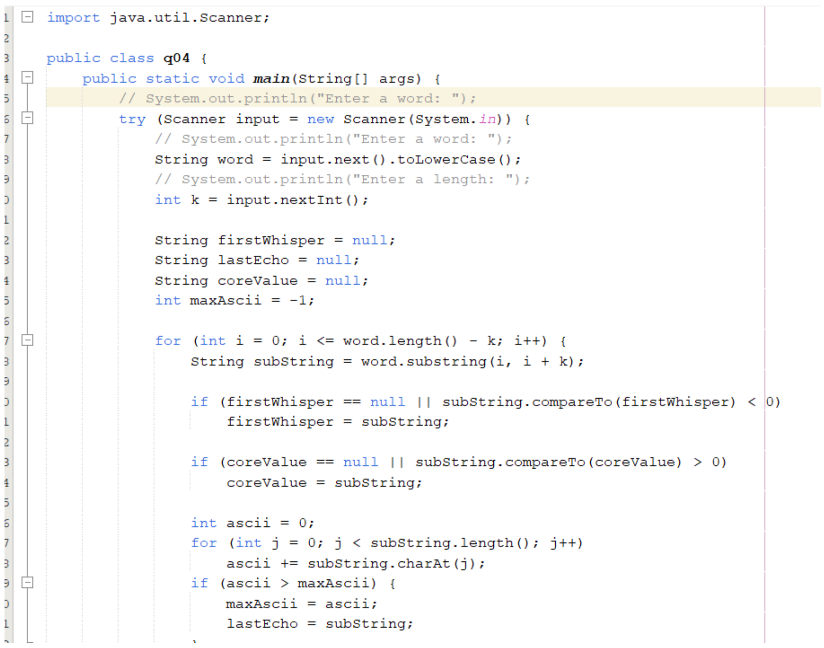
Input-

```
Satayisverysedap
3
```

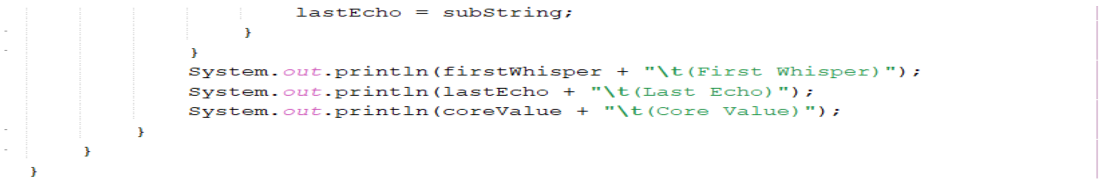
Output-

```
ata
rys
yse
```

4)Source Code



```java
import java.util.Scanner;

public class q04 {
    public static void main(String[] args) {
        // System.out.println("Enter a word: ");
        try (Scanner input = new Scanner(System.in)) {
            // System.out.println("Enter a word: ");
            String word = input.next().toLowerCase();
            // System.out.println("Enter a length: ");
            int k = input.nextInt();

            String firstWhisper = null;
            String lastEcho = null;
            String coreValue = null;
            int maxAscii = -1;

            for (int i = 0; i <= word.length() - k; i++) {
                String subString = word.substring(i, i + k);

                if (firstWhisper == null || subString.compareTo(firstWhisper) < 0)
                    firstWhisper = subString;

                if (coreValue == null || subString.compareTo(coreValue) > 0)
                    coreValue = subString;

                int ascii = 0;
                for (int j = 0; j < subString.length(); j++)
                    ascii += subString.charAt(j);
                if (ascii > maxAscii) {
                    maxAscii = ascii;
                    lastEcho = subString;
```



```java
                    lastEcho = subString;
                }
            }
            System.out.println(firstWhisper + "\t(First Whisper)");
            System.out.println(lastEcho + "\t(Last Echo)");
            System.out.println(coreValue + "\t(Core Value)");
        }
    }
}
```

Question 5: Uncle Lim's Golden Harmony Lanterns

Maintainer: 25065764   CHENYUHAN

1) problem.

Uncle Lim has a rule names “Golden Harmony” for the words painted on his lanterns.

The conditions of rule is

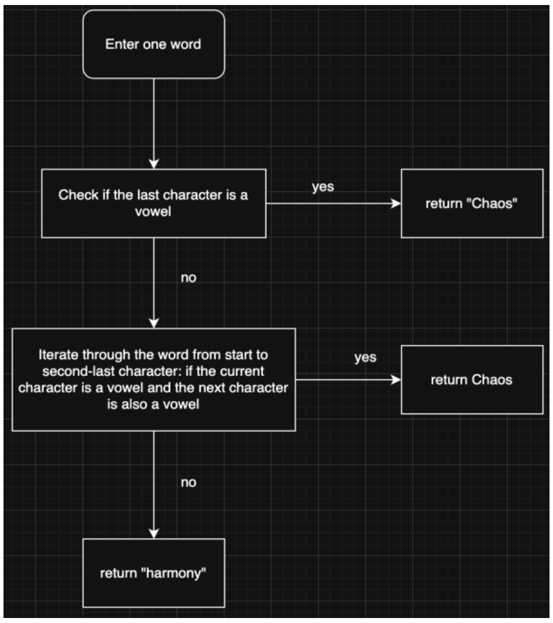
·Vowel can’t be the last letter of the word

·Vowel can’t be followed by another vowel

Vowel: (a e i o u)

If the word breaks one or all of rules, it’s considered as “Chaos”. Otherwise it’s “Harmony”.

2) Solution


Enter one word
Check if the last character is a vowel
yes
return "Chaos"
no
Iterate through the word from start to second-last character: if the current character is a vowel and the next character is also a vowel
yes
return Chaos
no
return "harmony"

Check if the last character is a vowel. If yes, return "Chaos".

a. Iterate through the word from start to second-last character:

b. · If the current character is a vowel and the next character is also a vowel, return "Chaos".

c. If neither rule is broken, return "Harmony".

3) Sample Input and Output.

input:

5

syntaxr   good   bachelor   demonstration   gembira

output:

Harmony   Chaos   Harmony   Chaos   Chaos

4) Source code

```java
import java.util.Scanner;

public class q05 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // System.out.print("Enter the number of words you are entering: ");
        int t = input.nextInt();
        String[] word = new String[t];
        for (int i = 0; i < t; i++) {
            // System.out.print("Enter word " + (i + 1) + ": ");
            word[i] = input.next().toLowerCase();
        }
        for (int i = 0; i < t; i++) {
            boolean notEndWithVowel = true;
            boolean noAdjacentVowel = true;
            char cEnd = word[i].charAt(word[i].length() - 1);
            for (int j = 0; j < word[i].length() - 1; j++) {
                char c1 = word[i].charAt(j);
                char c2 = word[i].charAt(j + 1);
                boolean c1Vowel = false;
                boolean c2Vowel = false;
                if (c1 == 'a' || c1 == 'e' || c1 == 'i' || c1 == 'o' || c1 == 'u') c1Vowel = true;
                if (c2 == 'a' || c2 == 'e' || c2 == 'i' || c2 == 'o' || c2 == 'u') c2Vowel = true;
                if (c1Vowel == true && c2Vowel == true) noAdjacentVowel = false;
            }
            if (cEnd == 'a' || cEnd == 'e' || cEnd == 'i' || cEnd == 'o' || cEnd == 'u') notEndWithVowel = false;
            if (noAdjacentVowel == true && notEndWithVowel == true) System.out.print("Harmony ");
            else System.out.print("Chaos ");
        }
        System.out.println();
        input.close();
    }
}
```

Question 6: Alex's Stutter Decompression

Maintainer: 25071951 LI YUCHEN

1) Problem

- Decompress the log string to count its length

Rules:

Compressed "stutter" format:

- Letters = append directly
- Digits (2–9) = repeat previous letter (digit-1) more times

Invalid Conditions

- First char is a digit
- Digit follows another digit
- Contains 0 or 1

If valid - output decompressed length

If invalid - output "Invalid Log"

2) Solution

1) Get an integer, t, for the number of log strings to test
2) Get compressed log string in next t line
3) Start check each log string and calculate in a loop

   First initialize the log count. While meet any invalid condition, break the loop and set count to -1

   a) Extract 2 characters sequentially from the log
   b) Confirm first character c1 is not a digit, **if c1 is digit**, end the loop for this log.
   c) Then check character c2 is either letter or digit, **if c2 is digit**, add the digit to count and j++ to skip one loop
   d) Else **if c2 is a letter**, first check if it is the **last character** in the log, if true, add 2 to count for c1 and c2

      Else c2 is a letter and **not the last character** in log, just add 1 to count for c1

e) For any input other than digit or letter, end the loop for this log.

4) Output the result with a space after it, repeat t times.

3) Sample Input & Output

Input:

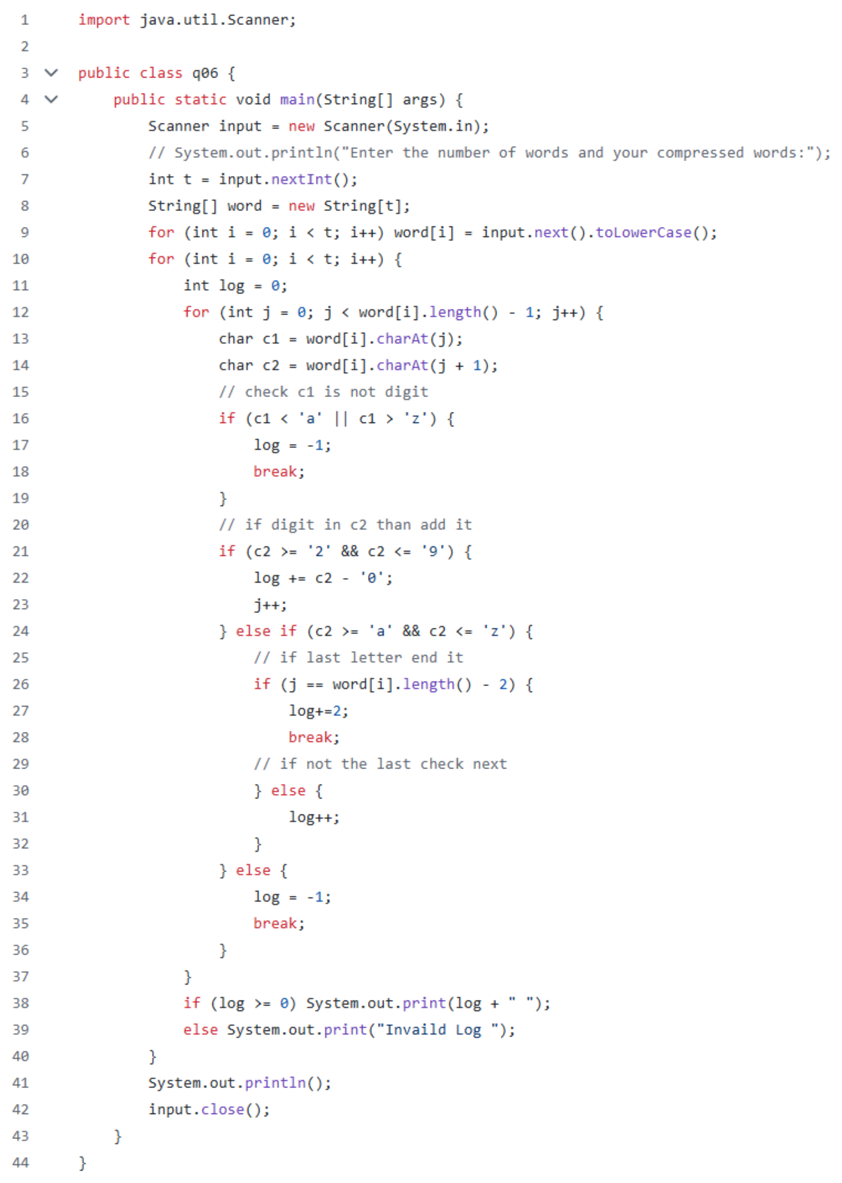
```
5 a4b2 log5 4bidden test1ng xy22z
```

Output:

```
6 7 Invalid Log Invalid Log Invalid Log
```

4) Source code

[WIX1002_Viva1/Viva1/src/q06.java at main · LiYuchen-UM/WIX1002_Viva1]



```java
import java.util.Scanner;

public class q06 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);
        // System.out.println("Enter the number of words and your compressed words:");
        int t = input.nextInt();
        String[] word = new String[t];
        for (int i = 0; i < t; i++) word[i] = input.next().toLowerCase();
        for (int i = 0; i < t; i++) {
            int log = 0;
            for (int j = 0; j < word[i].length() - 1; j++) {
                char c1 = word[i].charAt(j);
                char c2 = word[i].charAt(j + 1);
                // check c1 is not digit
                if (c1 < 'a' || c1 > 'z') {
                    log = -1;
                    break;
                }
                // if digit in c2 than add it
                if (c2 >= '2' && c2 <= '9') {
                    log += c2 - '0';
                    j++;
                } else if (c2 >= 'a' && c2 <= 'z') {
                    // if last letter end it
                    if (j == word[i].length() - 2) {
                        log+=2;
                        break;
                    // if not the last check next
                    } else {
                        log++;
                    }
                } else {
                    log = -1;
                    break;
                }
            }
            if (log >= 0) System.out.print(log + " ");
            else System.out.print("Invaild Log ");
        }
        System.out.println();
        input.close();
    }
}
```