
UNIVERSITY OF MALAYA


WIX1002 : FUNDAMENTALS OF PROGRAMMING

GROUP MEMBERS :
JITESH A/L MOGANA RAJA (25006745)
TAN CHEE KEAT (25006123)
TEH XU ZHE (25006355)
LEE MING DAO (25006825)
LIM HONG ZHANG (25006100)

FACULTY: FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY

TITLE: VIVA 1 REPORT

SEMESTER SESSION: SEM I 2025/2026

LECTURER NAME : DR. NURUL BINTI JAPAR

GROUP NAME : Ctrl+C Ctrl+V

**QUESTION 1**

**1.1 Problem**

The problem requires calculating a sequence of numbers (a "charm") for a given number of inquiries (q). For each inquiry, it contains an Initial Value (a), Multiplier Seed (b), and Charm Length (n). A charm sequence with **n** numbers will be generated following the calculation rule below:

$$\mathbf{Ci = a + (b \times 2^{i})}$$

i = The index in the sequence for the charm formed ( i starts at 0 and goes up to n - 1)

**1.2 Solution**

**1.2.1 IPO:**

1. **INPUT :**

- An integer q (total number of inquiries).

- Parameters for the q inquiries:
    - Integer a (Initial Value)
    - Integer b (Multiplier Seed)
    - Integer n (Charm Length)

2. **PROCESS :**

   The program iterates through each of the q inquiries, processing new values for a, b, and n for every iteration.

   For a single inquiry:

- Scan the charm length from step j =1 up to n
    - Calculate the specific charm: ans= a + (b x 2^(j-1))

- After the inner loop is finished, print a new line for the next inquiry.

3. **OUTPUT :**

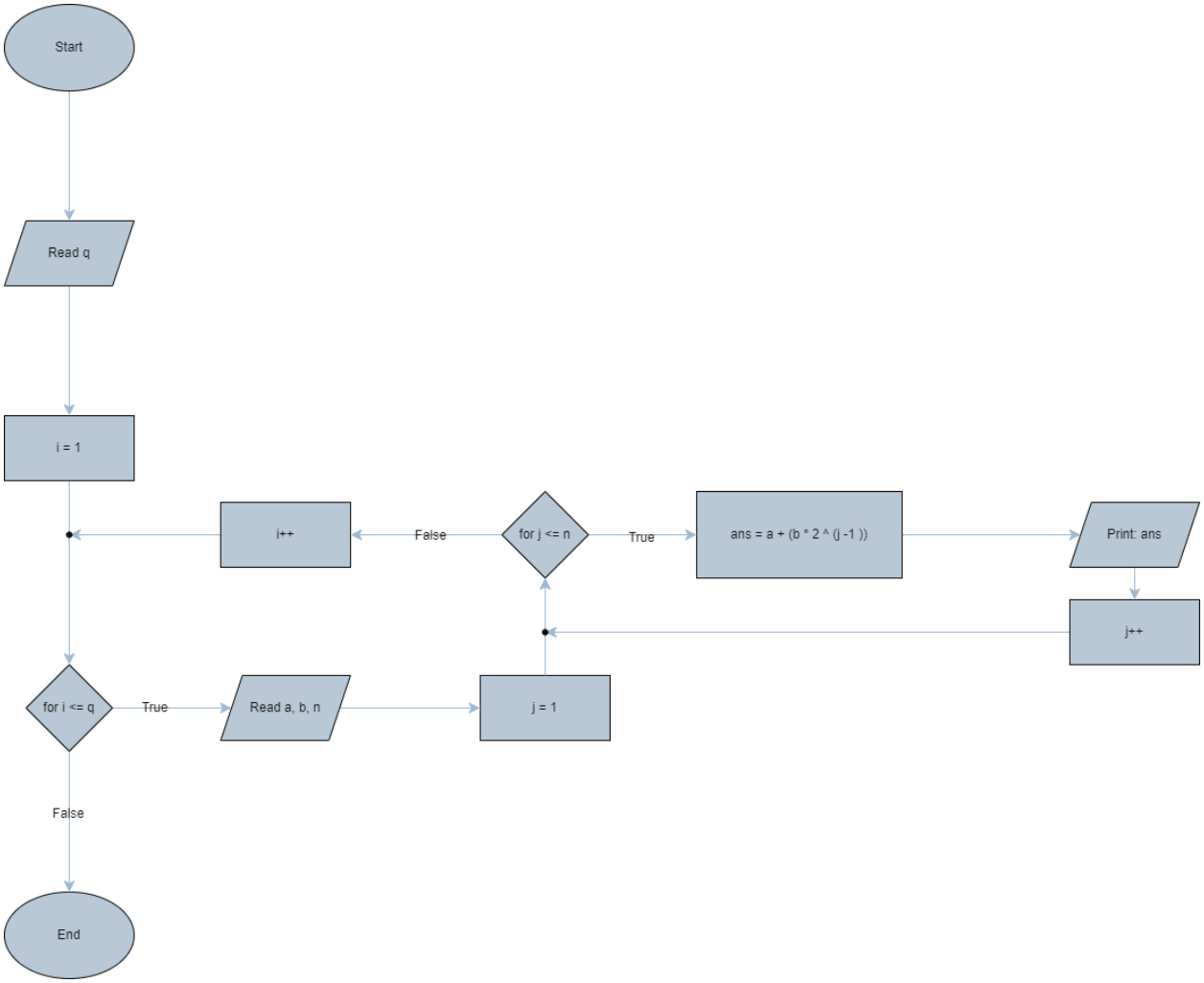
   For each inquiry:

- A sequence of n integers (the charm numbers) separated by space is generated.

**1.2.2 Pseudocode:**

```
Start
    Display: “Please enter the number of inquiries:”
    Read q
    For each inquiry from 1 to q:
        Display:  "Please enter the %d queries in the format: [Initial Value] [Multiplier
Seed][Charm Length]"
        Read a, b, n
        j=1
        For j to n:
            ans = a + (b * (2 ^ (j - 1)))
            Display: ans + “  ”
        End for
        Display: New line
    End for
    Close input
End
```

### 1.2.3 Flow Chart:



### 1.3 Sample Input and Output

```
PS C:\Users\mingl\ComputerScience\FOP\Viva_1>
PS C:\Users\mingl\ComputerScience\FOP\Viva_1>  c:; cd 'c:\Users\mingl\ComputerScience\FOP\Viv
wCodeDetailsInExceptionMessages' '-cp' 'C:\Users\mingl\AppData\Roaming\Code\User\workspaceSto
040d5\bin' 'q1'
Please enter the number of inquiries:
2
Please enter the 1 queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]
12 6 8
18 24 36 60 108 204 396 780
Please enter the 2 queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]
17 15 6
32 47 77 137 257 497
PS C:\Users\mingl\ComputerScience\FOP\Viva_1>
```

```
Please enter the number of inquiries:
3
Please enter the 1 queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]
4 6 14
10 16 28 52 100 196 388 772 1540 3076 6148 12292 24580 49156
Please enter the 2 queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]
23 45 12
68 113 203 383 743 1463 2903 5783 11543 23063 46103 92183
Please enter the 3 queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]
1 2 3
3 5 9
PS C:\Users\mingl\ComputerScience\FOP\Viva_1>
```

### 1.4 Source Code

```
import java.util.Scanner;

public class Q1 {
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);
        int q;

        System.out.println("Please enter the number of inquiries:");
        q = input.nextInt();

        for (int i = 1; i <= q; i++) {
```

```
        int a, b, n;
        System.out.printf("Please enter the %d queries in the format: [Initial Value] [Multiplier Seed] [Charm Length]\n", i);

        a = input.nextInt();
        b = input.nextInt();
        n = input.nextInt();

        int ans;

        for (int j = 1; j <= n; j++) {
            ans = (int) (a + (b * (Math.pow(2, j - 1))));
            System.out.print(ans + " ");
        }
        System.out.println();
    }
    input.close();
  }
}
```

**Question 2**

**2.1 Problem**

Ah Hock determines the "Digital Signature" of a number based on a Lucky Digit.

Given a number analyzeDigit , each digit is categorized in this priority:

1. Lucky Digit (digit = LuckyDigitValue)
2. Zero (digit = 0, only if LuckyDigitValue ≠ 0)
3. Even digits (2, 4, 6, 8)
4. Odd digits (1, 3, 5, 7, 9)

After counting:

- If LuckyCount is highest → LUCKY
- If EvenCount is highest → BALANCED
- If OddCount is highest → ENERGETIC
- Otherwise → NEUTRAL (includes ties and Zero being highest)

Special rules:

- If analyzeDigit = 0 → treat as a single digit 0
- If Lucky Digit = 0 → zeros count as Lucky, not Zero

  The program reads multiple test cases and prints the correct signature for each.

**2.2 Solution**

**2.2.1 IPO**

Input

- Q: number of test cases

  For each test case:
- analyzeDigit: the number to analyze
- LuckyDigit: the Lucky Digit

Process

1. For each test case, set counters to 0: Lucky, Zero, Even, Odd.
2. If analyzeDigit = 0, treat it as one digit.
3. Extract each digit of analyzeDigit using modulus and division.
4. Categorize each digit in this order:

    - If digit equals L → Lucky++
    - Else if digit equals 0 → Zero++
    - Else if digit is even → Even++
    - Else → Odd++

5. Compare counts:
    - If Lucky is strictly highest → “LUCKY”
    - Else if Even is strictly highest → “BALANCED”
    - Else if Odd is strictly highest → “ENERGETIC”
    - Else → “NEUTRAL”

Output

- The Digital Signature for each test case:

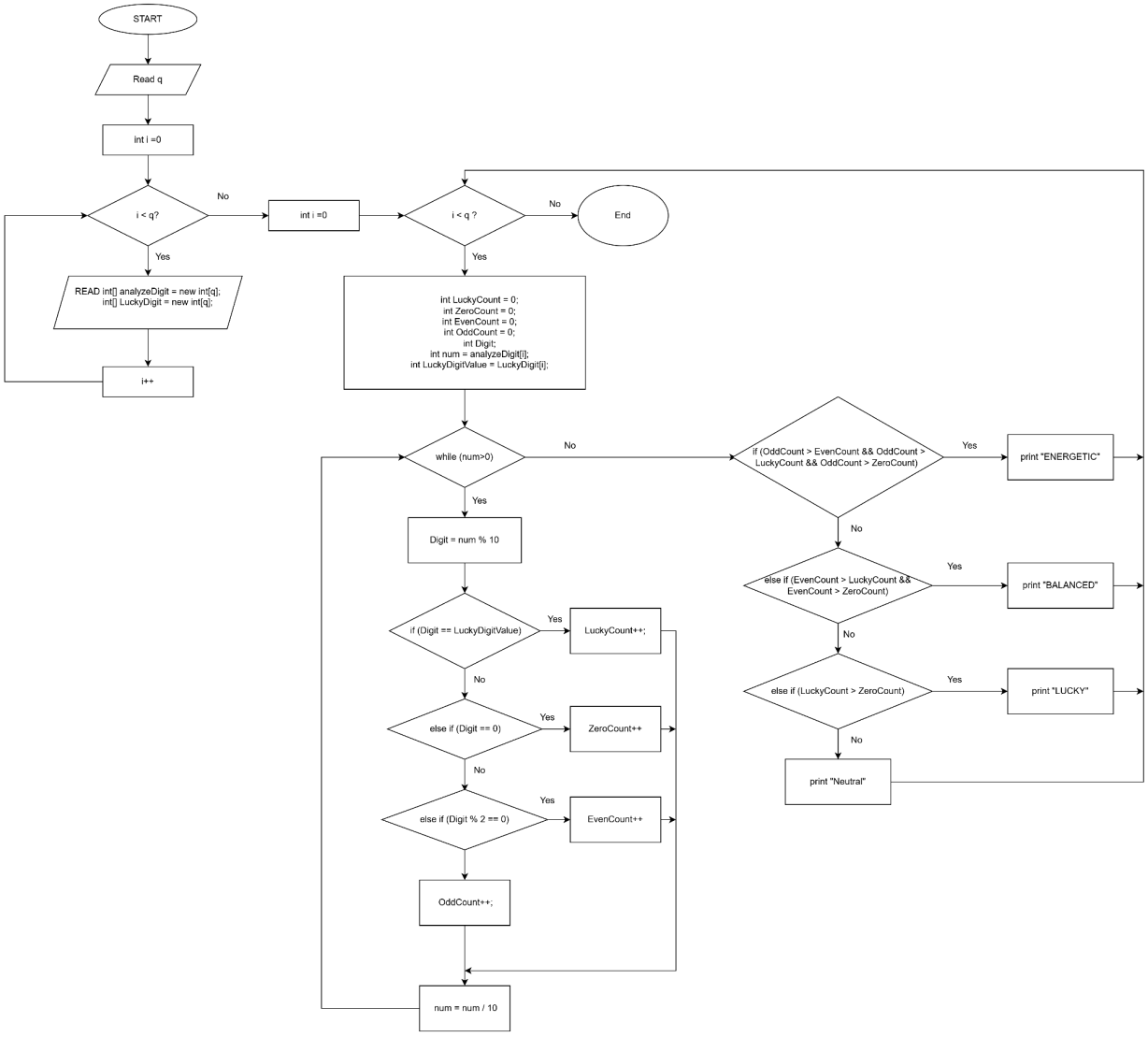
  LUCKY, BALANCED, ENERGETIC, or NEUTRAL

**2.2.2 Pseudocode**

```
START
    READ q  // number of test cases
    SET i = 0
    FOR i < q
        READ analyzeDigit , LuckyDigit
    END FOR
    SET i = 0
    FOR i < q
        // Initialize counter
        SET luckyCount = 0
        SET zeroCount = 0
        SET evenCount = 0
        SET oddCount = 0
        SET int Digit;
        SET int num = analyzeDigit[i];
        SET int LuckyDigitValue = LuckyDigit[i];
        // Process each digit (works even if N = 0)
        WHILE num>0
            SET Digit = analyzeDigit % 10

             // Categorize digit
             IF Digit == LuckyDigitValue THEN
                LuckyCount = luckyCount + 1
            ELSE IF Digit == 0 THEN
                ZeroCount = zeroCount + 1
             ELSE IF digit % 2 == 0 THEN
                EvenCount = EvenCount + 1
            ELSE
                OddCount = OddCount + 1
            END IF
```

```
            Num = Num / 10
        END WHILE
        // Determine signature
        IF (OddCount > EvenCount && OddCount > LuckyCount && OddCount >
    ZeroCount)
            PRINT “ENERGETIC”
        ELSE IF  (EvenCount > LuckyCount && EvenCount > ZeroCount)
            PRINT “BALANCED”
        ELSE IF (LuckyCount > ZeroCount)
            PRINT “LUCKY”
         ELSE
            PRINT “NEUTRAL”
        END IF
    END FOR
END
```

# 2.2.3 Flowchart



## 2.3 Sample input and output

Sample input:

```
4
881307 8
2213 5
1110 1
8888 8
```

Sample output:

```
ENERGETIC
BALANCED
LUCKY
LUCKY
```

## 2.4 Source Code

```
import java.util.Scanner;
public class Q2 {

    public static void main(String[] args) {
        // Setup object import
        Scanner sc = new Scanner(System.in);
        // Receive the new inputs through an integer q
        int q = sc.nextInt();

        // Create two arrays to store the analyzeDigit and LuckyDigit values
        int[] analyzeDigit = new int[q];
        int[] LuckyDigit = new int[q];

        // Loop to store the inputs in the respective arrays
        for (int i = 0; i < q; i++) {
            analyzeDigit[i] = sc.nextInt(); // Input for analyzeDigit
            LuckyDigit[i] = sc.nextInt(); // Input for LuckyDigit
        }
        // Loop to analyze each digit in the analyzeDigit array
        for (int i = 0; i < q; i++) {
            // Initialize counters for Lucky, Zero, Even, and Odd digits
            int LuckyCount = 0;
            int ZeroCount = 0;
            int EvenCount = 0;
            int OddCount = 0;
            int Digit;
            int num = analyzeDigit[i];
            int LuckyDigitValue = LuckyDigit[i];

            // Analyze each digit of the current analyzeDigit number
```

```
        while (num > 0) {
            // Extract the last digit through modulus operation
            Digit = num % 10; // Get the last digit
            // Check and increment the respective counters based on the digit value
            if (Digit == LuckyDigitValue) {
                LuckyCount++;
            } else if (Digit == 0) {
                ZeroCount++;
            } else if (Digit % 2 == 0) {
                EvenCount++;
            } else {
                OddCount++;
            }
            num = num / 10;
        }

        // Determine and print the classification based on the counts
        if (OddCount > EvenCount && OddCount > LuckyCount && OddCount > ZeroCount) {
            System.out.println("ENERGETIC");
        } else if (EvenCount > LuckyCount && EvenCount > ZeroCount) {
            System.out.println("BALANCED");
        } else if (LuckyCount > ZeroCount) {
            System.out.println("LUCKY");
        } else {
            System.out.println("NEUTRAL");
        }

    }

  }
}
```

**QUESTION 3**

**3.1 Problem**

The problem requires us to write a program that reads an integer T, the number of test cases. For each test case, read an integer H (Height) and a character S (Style).

Based on the style S, print the corresponding pattern of height H:

1. Style 'A' (Angled): Print H rows. Row i (from 1 to H) must contain the digit i repeated i times.
   Example (H=3):

   1

   22

   333

2. Style 'P' (Pyramid): Print H rows. Row i must be a centered, palindromic number sequence counting from 1 up to i and back down to 1.
   Example (H=3):

     1

    1 2 1

   1 2 3 2 1

### 3.2 Solution

#### 3.2.1 IPO Chart

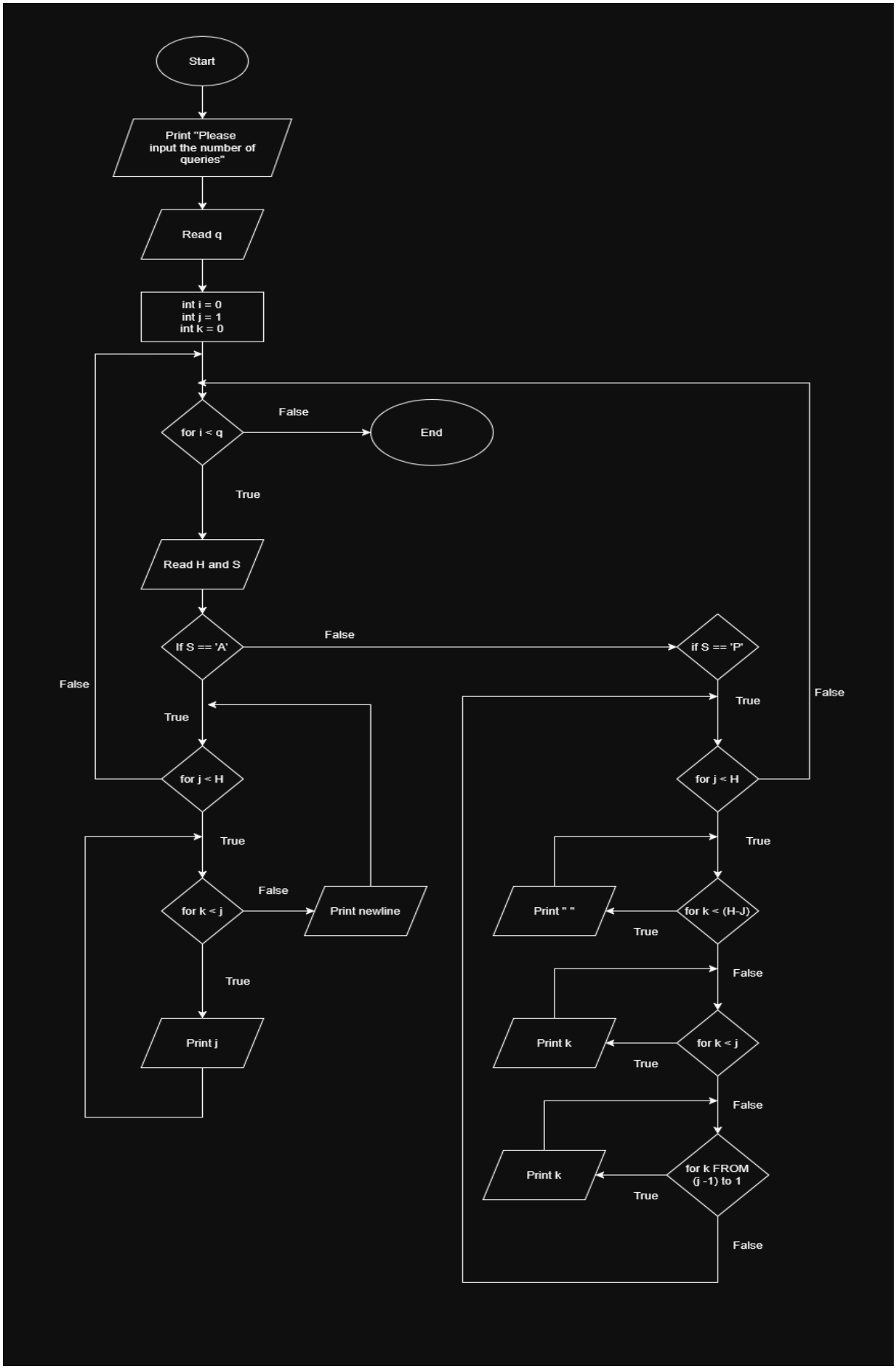
| Input | Process | Output |
|---|---|---|
| - T (an integer for the number of test cases)<br>- For each test case:<br>  - H (an integer for the pattern's height)<br>  - S (a character for the style, 'A' or 'P' | 1. Read the number of test cases, T.<br>2. Start a loop that repeats T times.<br>3. Inside the loop, read the height (H), and the style (S).<br>4. Check the value of S:<br>  a. If S is 'A':<br>  i. Start an outer loop for rows (we use j) from 1 - H.<br>  ii. Start an inner loop from 1 - j.<br>  iii. Print the value of j.<br>  iv. After the inner loop, print a new line.<br>  b. If S is 'P':<br>  i. Start an outer loop for rows (we use j) from 1 - H.<br>  ii. Print (H - j) spaces.<br>  iii. Print numbers from 1 up to j.<br>  iv. Print numbers from j - 1 to 1<br>  v. Print a new line.<br>5. Repeat the loop until all T test cases are completed. | - The complete number pattern for each test case. |

### 3.2.2 Pseudocode

```
START
    PRINT “Please input the number of queries: “
    READ q
    FOR i < q
        READ H
        READ S
        IF S equals ‘A’ THEN
            FOR j FROM 1 TO H
                FOR k FROM 1 to j
                    PRINT j
                END FOR
                PRINT newline
            END FOR
        END IF

        IF S equals ‘P’ THEN
            FOR j FROM 1 TO H
                FOR k < (H - j)
                    PRINT “ “
```

```
                END FOR
                FOR k FROM 1 to j
                    PRINT k
                END FOR
                FOR k FROM (j - 1) TO 1
                    PRINT k
                END FOR
                PRINT newline
            END FOR
        END IF
    END FOR
END
```

### 3.2.3 Flow Chart



### 3.3 Sample Input and Output

**Sample Input**

```
run:
Please input the number of queries: 3
Please input the height of the pattern and the style (A or P) separated by spaces:
4 P 5 A 3 P
```

**Sample Output**

```
    1
   121
  12321
 1234321

1
22
333
4444
55555

  1
 121
12321
```

**3.4 Source Code**

```
package q3;

import java.util.Scanner;

/**
 *
 * @author ekitstrap
 */
public class Q3 {

    public static void main(String[] args) {
        // Setup object imports and variables
        Scanner input = new Scanner(System.in);

        // Receive the number of queries (q) first
        System.out.print("Please input the number of queries: ");
        int q = input.nextInt();

        System.out.println("Please input the height of the pattern and the style (A or P) separated by
spaces: ");

        // Repeat for every number of query
        for (int i = 0; i < q; i++) {
            int H = input.nextInt();
            char S = input.next().charAt(0);

            // Making it a character object to use the equals function
            // Check for the pattern the user wants (tip: confirmed uppercase character)
            // Angled staircase (A)
            if (S == 'A') {
                for (int j = 1; j <= H; j++) {
                    for (int k = 1; k <= j; k++) {
                        System.out.print(j);
                    }
                    System.out.println();
                }
            }
            // Pyramid
```

```
        if (S == 'P') {
            // Setting the numbers
            for (int j = 1; j <= H; j++) {

                // Setting up the spacing
                for (int k = 0; k < (H - j); k++) {
                    System.out.print(" ");
                }

                // Ascending loop -> to reach the max value
                for (int k = 1; k <= j; k++) {
                    System.out.print(k);
                }

                // Descending loop -> to reach 1
                for (int k = j - 1; k >= 1; k--) {
                    System.out.print(k);
                }

                System.out.println();
            }

        }
        System.out.println();
    }
  }

}
```

## QUESTION 4

### 4.1 Problem

The question asks us to process a given word and identify three special “word gems” inside it based on a specified substring length. We are required to examine every possible substring of that length and determine three results:

1. First Whisper – the substring that is lexicographically smallest.
2. Last Echo – the substring that is lexicographically largest.
3. Core Value – the substring with the highest total ASCII value.
   If two have the same value, the one that appears first is chosen.

The program must read the word and convert it into lowercase before processing. It will then generate all substrings of the given length, compare them accordingly, and output the three results, each on a separate line.

**4.2 Solution**

1. Read input from the user
   a. Ask the user to enter a word and convert it to lowercase.
   b. Ask the user to enter the substring length *k*.

2. Initialise starting values
   a. Extract the first substring of length *k*.
   b. Set this substring as the initial firstWhisper, lastEcho, and coreValue.
   c. Calculate the ASCII sum of this substring and store it as maxAsciiSum.

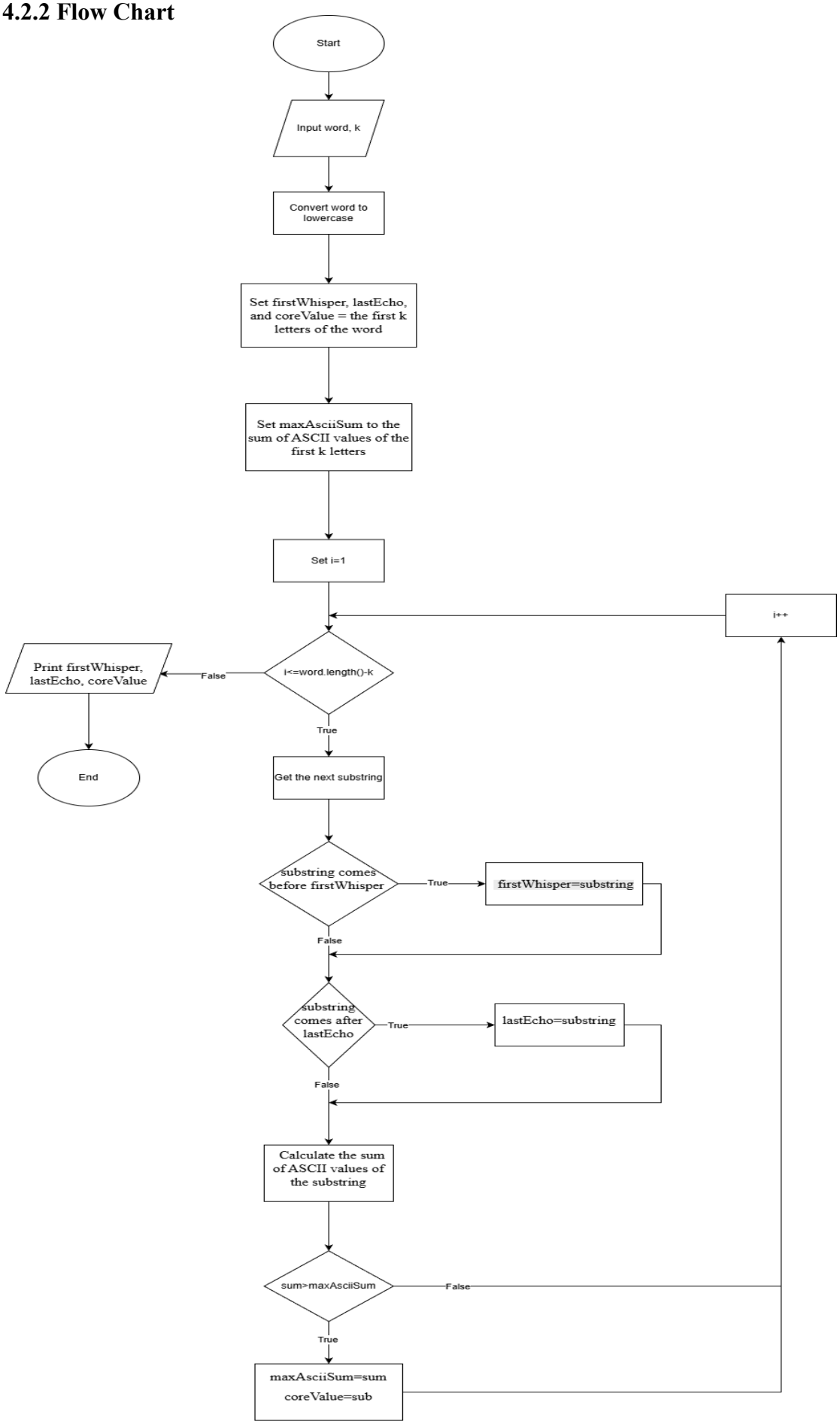
3. Generate and compare remaining substrings
   a. Loop through the word to extract each substring of length *k*.
   b. For each substring:
   – Update firstWhisper if it is lexicographically smaller.
   – Update lastEcho if it is lexicographically larger.
   – Calculate its ASCII sum and update coreValue if the sum is greater than maxAsciiSum.

4. Display all results
   a. Print firstWhisper, lastEcho, and coreValue.

### 4.2.1 Pseudocode

```
START
    INPUT word
    CONVERT word to lowercase

    INPUT length of substrings, k
    SET firstWhisper, lastEcho, and coreValue = the first k letters of the word
    SET maxAsciiSum to the sum of ASCII values of the first k letters

    FOR each substring of length k in the word starting from the second character:

        READ the substring

        IF the substring comes before firstWhisper lexicographically:
            firstWhisper=substring
        END IF
        IF the substring comes after lastEcho lexicographically:
            lastEcho=substring
        END IF

        CALCULATE the sum of ASCII values of the substring
        IF this sum is greater than maxAsciiSum:
            maxAsciiSum=sum
            coreValue=sub
        END IF

    END FOR

    PRINT firstWhisper, lastEcho, coreValue

END
```

### 4.2.2 Flow Chart



**4.3 Sample Input and Output**

```
Enter the word: PenangLaksaPalingSedap
Enter the length of substrings, k: 4
First Whisper: aksa
Last Echo: seda
Core Value: ings


Enter the word: HelloWorldGoodbyeWorld
Enter the length of substrings, k: 3
First Whisper: bye
Last Echo: yew
Core Value: wor
```

### 4.4 Source Code

```
import java.util.Scanner;
public class Q4 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter the word: ");
      String word = sc.next().toLowerCase();
      System.out.print("Enter the length of substrings, k: ");
      int k = sc.nextInt();
      String firstWhisper, lastEcho, coreValue;
      firstWhisper = lastEcho = coreValue = word.substring(0, k);

      int maxAsciiSum = 0;
      for (int j = 0; j < coreValue.length(); j++) {
         maxAsciiSum += (int) coreValue.charAt(j);
      }

      for (int i = 1; i <= word.length() - k; i++) {
         String sub = word.substring(i, i + k);

         if (sub.compareTo(firstWhisper) < 0) {
            firstWhisper = sub;
         }

         if (sub.compareTo(lastEcho) > 0) {
            lastEcho = sub;
         }

         int sum = 0;
         for (int j = 0; j < sub.length(); j++) {
            sum += (int) sub.charAt(j);
         }

         if (sum > maxAsciiSum) {
            maxAsciiSum = sum;
            coreValue = sub;
         }
      }

      System.out.println("First Whisper: " + firstWhisper);
      System.out.println("Last Echo: " + lastEcho);
      System.out.println("Core Value: " + coreValue);

      sc.close();
   }
}
```

## Question 5

### 5.1 Problem

The problem requires building a validator program for Uncle Lim's "Golden Harmony" lantern collection. Each lantern contains exactly one word, and the word must follow two linguistic rules created by Uncle Lim. Any word that violates even one rule is labeled "Chaos", and only words that follow both rules are considered "Harmony".

The Golden Harmony rules are:

1. A vowel cannot be the last letter of the word.
   Vowels are: a, e, i, o, u.
2. A vowel cannot be immediately followed by another vowel.
   Any two consecutive vowels break this rule.

The program must read T words. For each word, it checks whether the word satisfies both rules. If the word passes both rules, print "Harmony". Otherwise, print "Chaos".

**5.2 Solution**

1. Read the number of words
   a. Ask the user to enter how many words will be checked.
   b. Store this value as T.

2. Prepare required data
   a. Create a string containing all vowels ("aeiou").
   b. Loop from 1 to T to process each word.

3. Process each word
   a. Read the current word and convert it into a character array.
   b. Assume the word is in harmony by setting isHarmony = true.

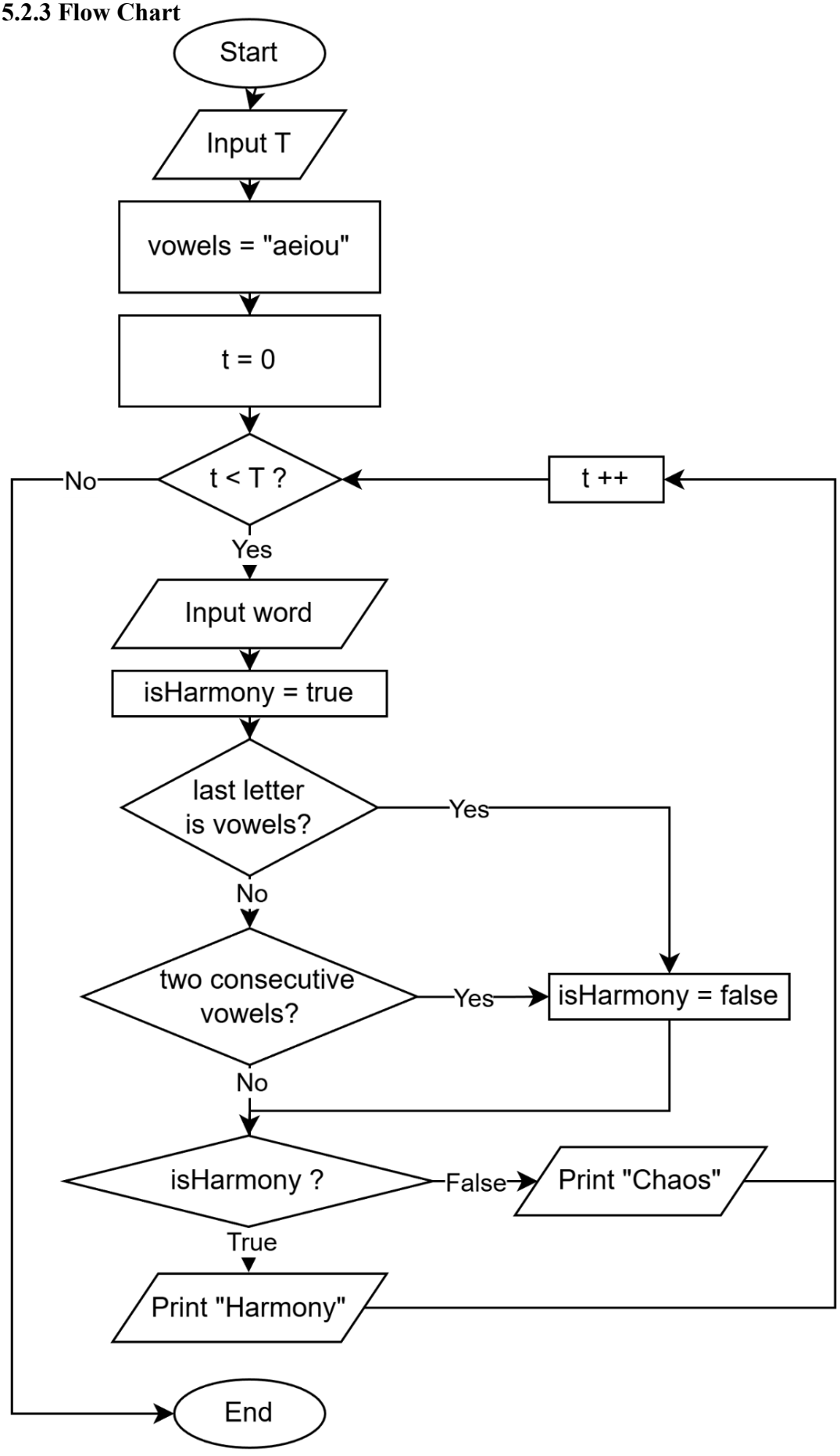
4. Check Harmony Rules
   a. Rule 1: The last letter must not be a vowel.
   – If the last character is found in the vowel list, set isHarmony = false.
   b. Rule 2: No two vowels can appear consecutively.
   – Loop through the word and check each pair of adjacent characters.
   – If both are vowels, set isHarmony = false and stop checking further.

5. Display the result
   a. If isHarmony is true, print "Harmony".
   b. Otherwise, print "Chaos".

**5.2.2 Pseudocode**

```
START
        INPUT T    // number of words
        SET vowels = "aeiou"

        FOR each word from 1 to T
                Input word
                isHarmony = true

        IF last letter of word is in vowels
                isHarmony = false
        END If

        FOR i from 0 to (length of letters - 2)
                IF letter[i] is vowel AND letter[i+1] is vowel, then
                        isHarmony = false
                        BREAK loop
                END If
        END FOR

        IF isHarmony == true
                PRINT "Harmony"
        ELSE
                PRINT "Chaos"
        END IF
        END FOR
END
```

**5.2.3 Flow Chart**



### 5.3 Sample Input and Output

```
Enter number of words: 3
Enter word 1: apple
  Result: Chaos
Enter word 2: coconut
  Result: Harmony
Enter word 3: orange
  Result: Chaos
```

```
Enter number of words: 4
Enter word 1: penang
  Result: Harmony
Enter word 2: laksa
  Result: Chaos
Enter word 3: paling
  Result: Harmony
Enter word 4: sedap
  Result: Harmony
```

**5.4 Source Code**

```
import java.util.Scanner;

public class Q5 {
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);

      System.out.print("Enter number of words: ");
      int T = sc.nextInt(); // number of words

      String vowels = "aeiou";
      for (int t = 0; t < T; t++) {
         System.out.printf("Enter word %d: ", t + 1);
         String word = sc.next();
         char[] letters = word.toCharArray();
         boolean isHarmony = true;

         // Rule 1: Last letter cannot be a vowel
         // indexOf returns -1 if the character is not found
         if (vowels.indexOf(letters[letters.length - 1]) != -1){
            isHarmony = false;
         }

         // Rule 2: No two consecutive vowels
         // Iterate through the word to check
         for (int i = 0; i < letters.length - 1; i++) {
            if (vowels.indexOf(letters[i]) != -1 &&
               vowels.indexOf(letters[i + 1]) != -1) {
               isHarmony = false;
               break;
            }
         }

         if (isHarmony) {
            System.out.println("  Result: Harmony");
         } else {
            System.out.println("  Result: Chaos");
         }
      }

      sc.close();
   }
}
```

**QUESTION 6**

**6.1 Problems**

The problem asks us to write a program that decompresses logs. The program will read a compressed string and determine:

- If the log is valid, output the length of the fully decompressed message.

- If the log is invalid, output "Invalid Log".

How the Decompression Works

You must read the compressed string from left to right.

1. Letters

- If you see a lowercase letter, add it directly to the decompressed length.

2. Digits (‘2’–‘9’)

- A digit means “repeat the previous character.”

- The digit d tells you to repeat the character right before it a total of d−1 more times.

Invalid Log Rules

A log becomes invalid immediately if:

i. The first character is a digit

ii. A digit appears right after another digit

iii. The log contains ‘0’ or ‘1’

**6.2 Solution**

**6.2.1 IPO**

INPUT:

-An integer T (number of compressed log strings).

-T compressed strings, each containing lowercase letters and digits and stored in an array.

PROCESS:

For each compressed string:

1. Initialise
   -length=0
   -isInvalid= false
2. Rule A — First Character Check
   -if the first character is a digit, set isInvalid= true
3. Scan the string from left to right
   -if current character is a letter
      -plus 1 to the length
   -if current character,d is a digit
      -Rule B — Previous Digit Check
      -if previous character is a digit, set isInvalid=true
      -Rule C — '0' and '1' Check
      -if current character is 0 or 1, set isInvalid=true
      -Otherwise, add d-1 to the length
4. If any rules break
   -Stop processing current string
   -Output "Invalid Log"

OUTPUT:

For each compressed string:

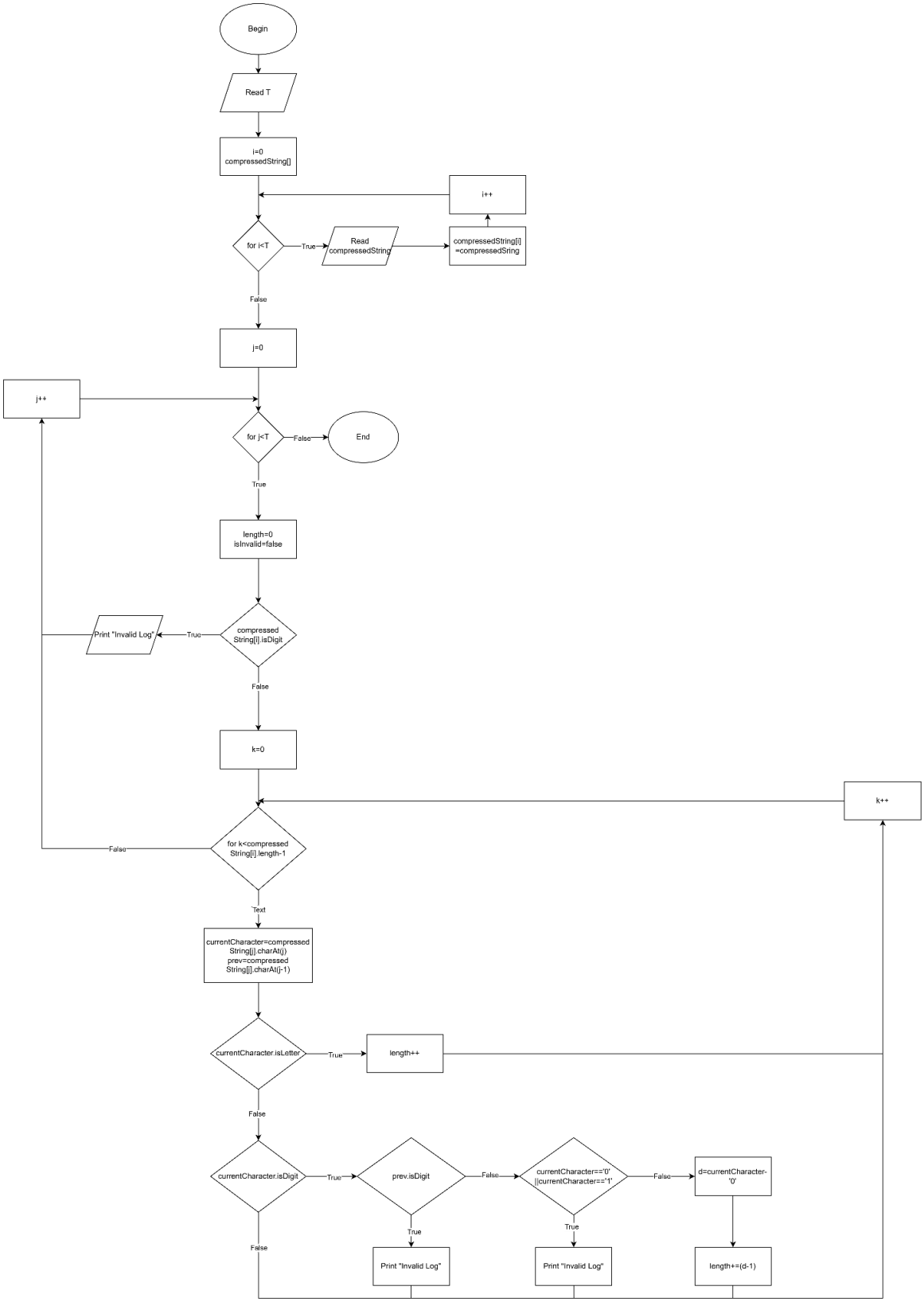
Either length or "Invalid Log"

### 6.2.2 Pseudocode

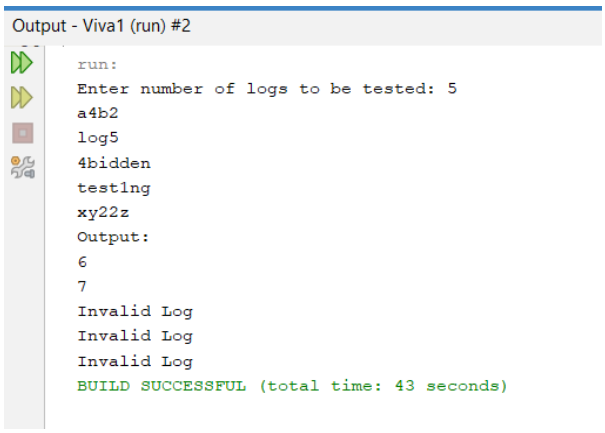
```
START

    READ T
    FOR i = 0 to T-1
        Read compressedString
        Store compressedString in array compressedString[i]
    END FOR

    FOR i = 1 to T
        SET length = 0
        SET isInvalid = false

        IF first character of compressedString is a digit
            PRINT "Invalid Log"
            CONTINUE to next string
        END IF
        FOR j = 0 to compressedString.length - 1

            currentChar = compressedString[j].charAt(j)

            IF currentChar is a letter
                length = length + 1

            ELSE IF currentChar is a digit

                IF currentChar == '0' or currentChar == '1'
                isInvalid = true
                break
                END IF
```

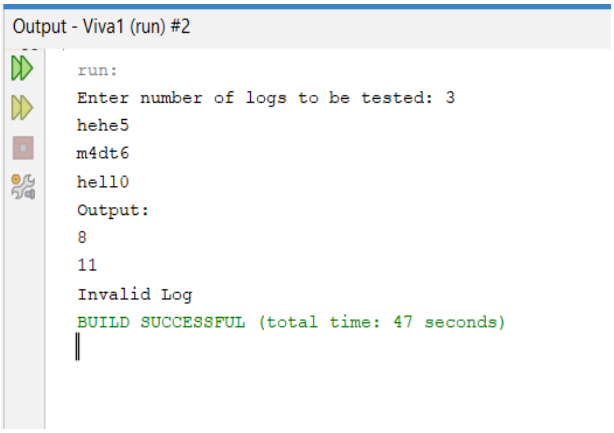
```
                    previousChar = compressedString[j - 1]

                    IF previousChar is a digit
                        isInvalid = true
                        break
                    END IF

                d = integer value of currentChar
                length = length + (d - 1)

                END IF
            END FOR

            IF isInvalid == true
                Print "Invalid Log"
            ELSE
                Print length
            END IF
        END FOR
END
```

## 6.2.3 Flowchart



### 6.3 Sample Input and Output




### 6.4 Source Code

```
import java.util.Scanner;
public class Q6 {
   public static void main(String[] args) {

      Scanner input= new Scanner (System.in);

      //Number of log strings to be tested
      System.out.print("Enter number of logs to be tested: ");
      int T=input.nextInt();

      //create an array of size T to store all the
      String[]compressed=new String[T];

      //loop for T times
      for (int i=0; i<T; i++){
         compressed[i]=input.next();
      }

      System.out.println("Output: ");
```

```
for(int i=0; i<T; i++){
    int length=0;
    boolean invalid = false;

    //rules A
    if(Character.isDigit(compressed[i].charAt(0))){
        System.out.println("Invalid Log ");
        continue;
    }

    //loop to check each character in th log
    for (int j=0; j<compressed[i].length();j++){

        //store current character
        char current= compressed[i].charAt(j);

        //current character is letter
        if(Character.isLetter(current)){
            length++;
        }
        //current character is digit
        else if(Character.isDigit(current)){

            //rules B
            if(current=='0'||current=='1'){
                invalid=true;
                break;
            }

            //rules C
```

```
                char prev= compressed[i].charAt(j-1);
                if (Character.isDigit(prev)){
                    invalid=true;
                    break;
                }

                //change char to int and add d-1 time to length
                int d=current-'0';
                length+=(d-1);

            }


        }
        if (invalid) {
            System.out.println("Invalid Log ");
        } else {
            System.out.println(length+ " ");
        }

    }
  }
}
```