PSEUDOCODE & FLOWCHART

# Reverse all the characters from a word in a string

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the input String:”

READ a line of text and STORE it in variable ‘input’

SPLIT ‘input’ into an array of words using space as delimiter

STORE this array in ‘array’

FOR each word in ‘array’ DO

CALL ‘reverseString’ method with the word as argument

END FOR

END

DEFINE ‘reverseString’ method with parameter ‘reverse’

FOR index ‘i’ from (length of ‘reverse’ - 1) down to 0 DO

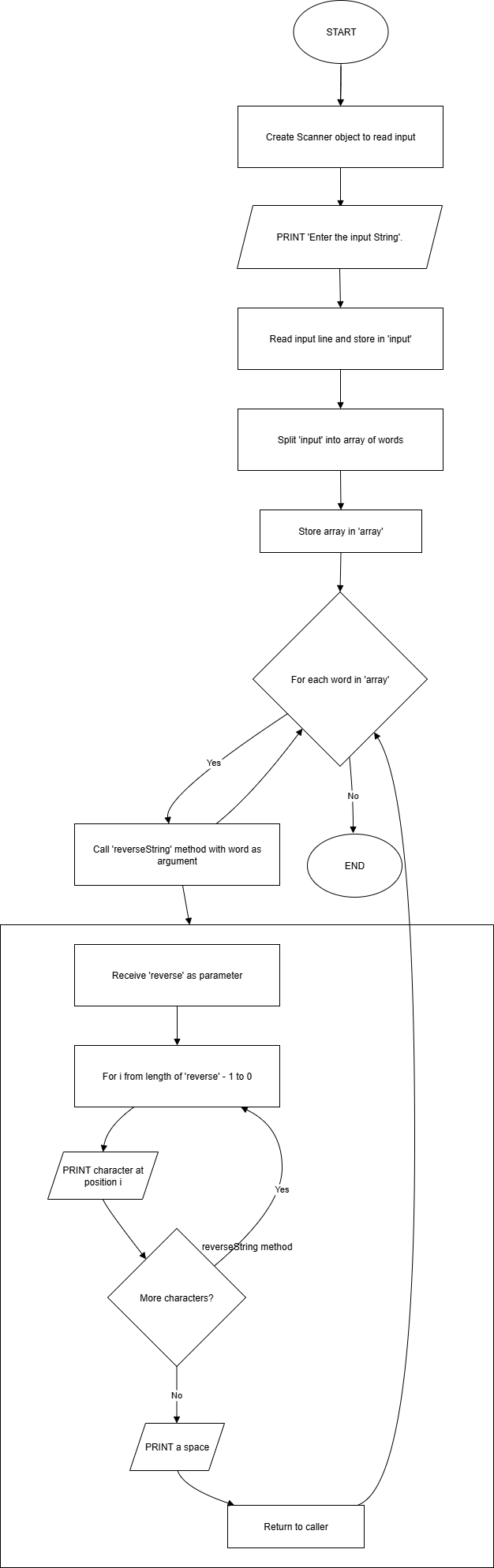
PRINT character at position ‘i’ in ‘reverse’

END FOR

PRINT a space

END

# FLOWCHART:



# Printing Armstrong numbers up to the range. And print the no. of values that are in the range.

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the input number:”

READ input number and STORE in variable ‘input’

INITIALIZE ‘count’ to 0

FOR each number ‘num’ from 1 to ‘input’ DO

INITIALIZE ‘sum’ to 0

STORE ‘num’ in ‘temp’

STORE ‘num’ in ‘n’

WHILE ‘n’ is greater than 0 DO

CALCULATE remainder ‘rem’ as ‘n’ modulo 10

ADD cube of ‘rem’ to ‘sum’

DIVIDE ‘n’ by 10

END WHILE

IF ‘sum’ is equal to ‘temp’ THEN

PRINT ‘temp’ // If Armstrong number found

INCREMENT ‘count’ by 1

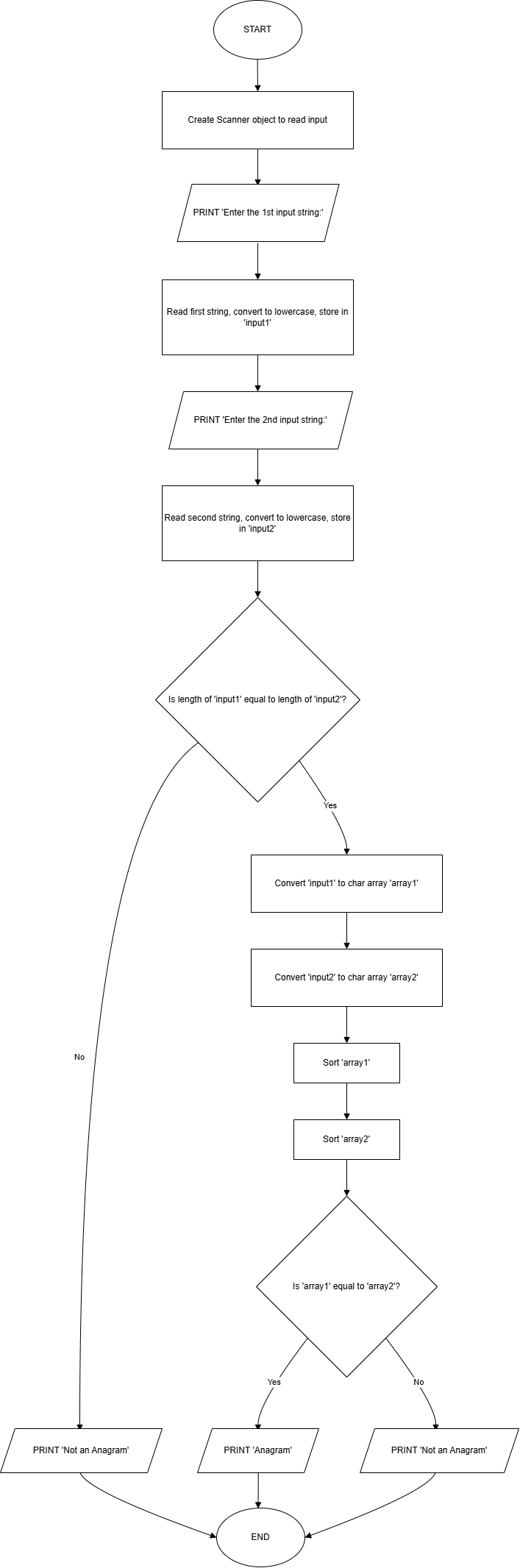
END IF

END FOR

PRINT “Total Armstrong numbers are:” followed by value of ‘count’

END

# FLOWCHART:



# Program for EMAIL VALIDATION using Pattern and Matcher classes in Regex. E.g. ([firstName.LastName1-10@enh.com](mailto:firstName.LastName1-10@enh.com) (or) .in)

# PESUDOCODE:

BEGIN

CREATE a Scanner object to rea input from user

PRINT “Enter the input String:”

READ input string and STORE in ‘input’

DEFINE ‘regex’ pattern as:

Starts with one or more alphabets (A-Z or a-z)

Followed by a dot ‘.’

Followed by a one or more alphabets (A-Z or a-z)

Followed by 1 to 10 digits

Followed by ‘@’

Followed by either ‘enh.com’ or ‘enh.in’

COMPILE the ‘regex’ into a ‘pattern’

MATCH the ‘input’ string against the ‘pattern’

IF input string matches the pattern, THEN

PRINT “Valid Email Address”

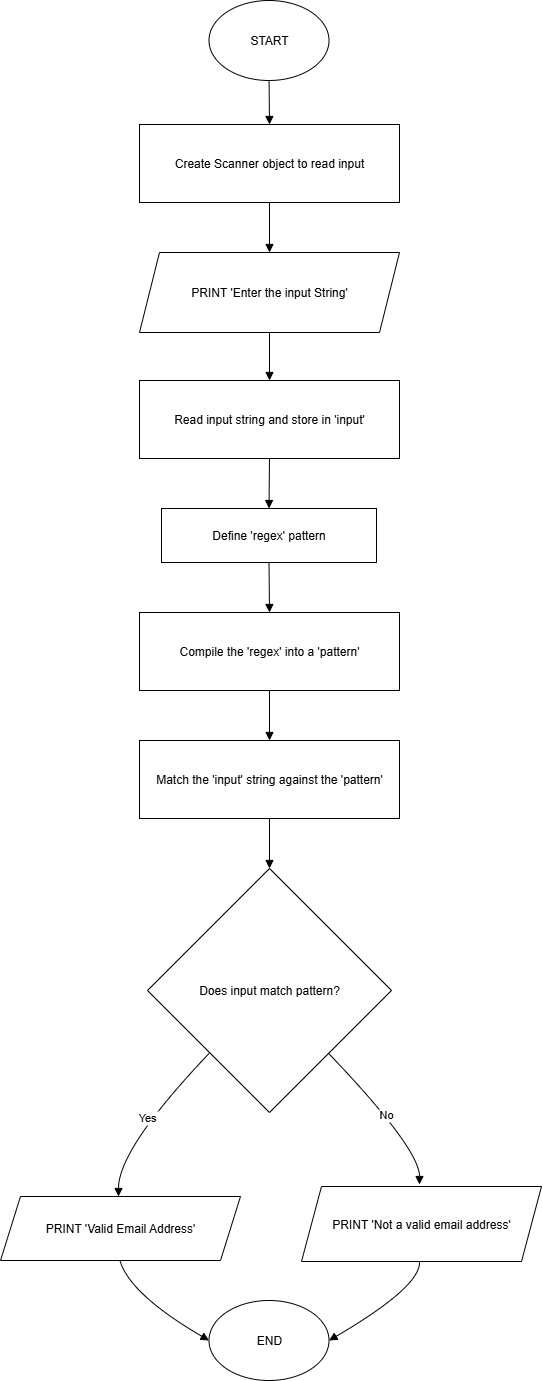
ELSE

PRINT “Not a valid email address”

END IF

END

# FLOWCHART:



# Program for Remove Duplicates from an array.

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the array size:”

READ array size and STORE in ‘size’

DECLARE an integer array ‘array’ of length ‘size’

PRINT “Enter” followed by ‘size’ number of elements

FOR index ‘i’ from 0 to (size - 1) DO

READ element and STORE in ‘array[i]’

END FOR

CONVERT ‘array’ to a Set ‘uniqueSet’

(This automatically removes duplicate values)

PRINT “After removing duplicates:”

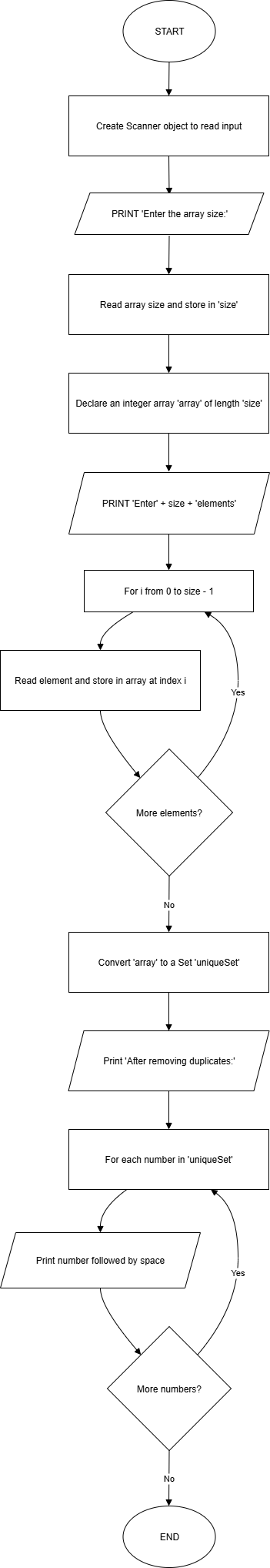
FOR each number in ‘uniqueSet’ DO

PRINT number followed by a space

END FOR

END

# FLOWCHART:



# Program for Remove Special characters from an array:

# PESUDOCODE:

BEGIN

PRINT “Enter input String:”

CREATE a Scanner object to read input from user

READ the input string and STORE in ‘input’

DEFINE a regex pattern ‘[^A-Za-z0-9]’

COMPILE the regex pattern into ‘pattern’

CREATE a ‘matcher’ to match the ‘input’ string against the ‘pattern’

­ REPLACE all occurrences matched by the ‘matcher’ with a space (“ ”)

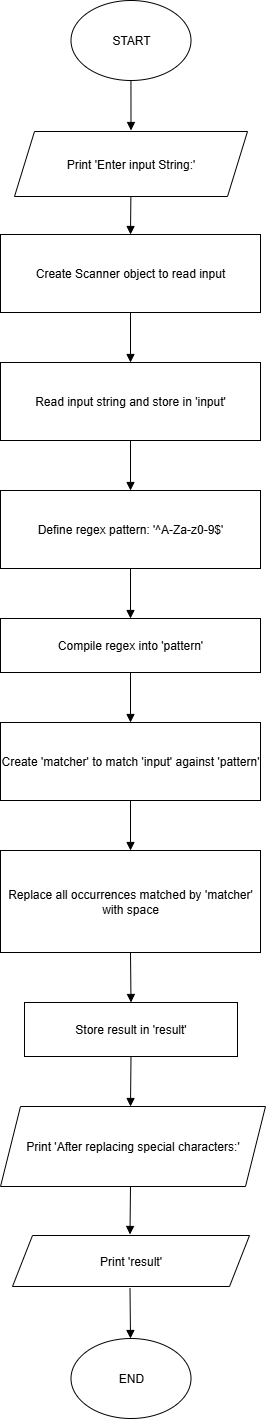
STORE the result in ‘result’

PRINT “After replacing special characters:”

PRINT ‘result’

END

# FLOWCHART:



# Program for rotate elements by positions to right and replaces the values with 0’s

# PESUDOCODE:

BEGIN

DECLARE and INITIALIZE an integer array ‘array’ with values {1,2,3,4,5,6,7}

SET variable ‘shift’ to 3

FOR ‘range’ from (length of ‘array’ - 1) down to 0 DO

IF ‘range’ is greater than or equal to ‘shift’ THEN

SET ‘array[range]’ to ‘array[range-shift]’

ELSE

SET ‘array[range]’ to 0

END IF

END FOR

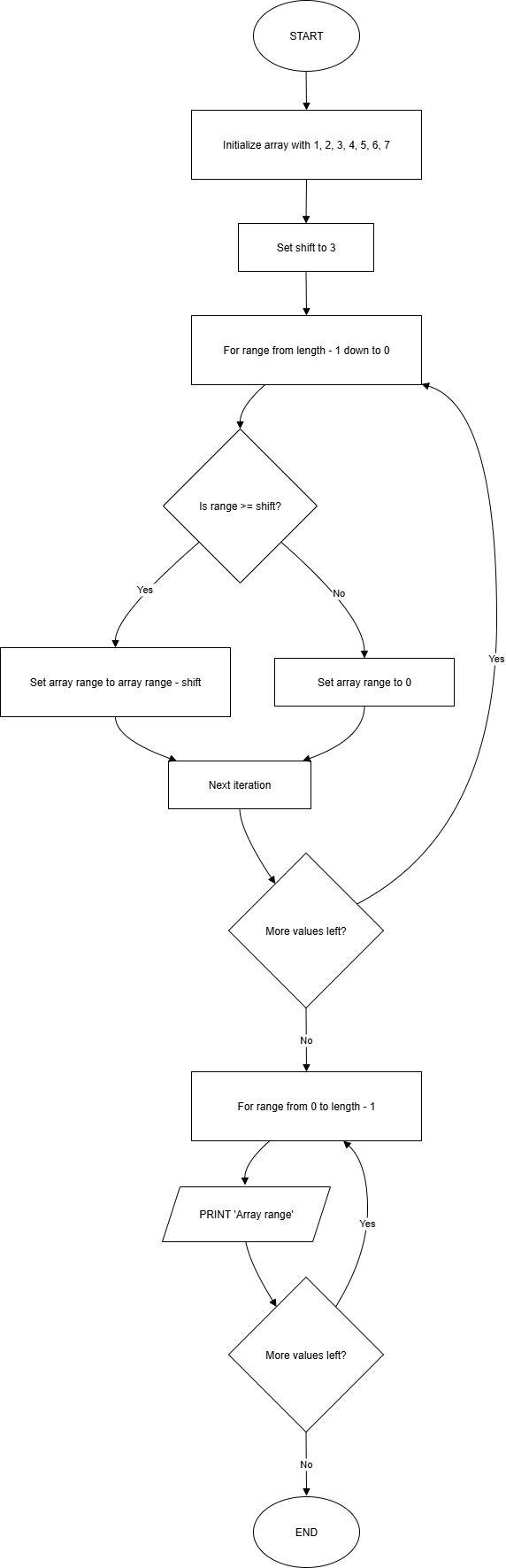
FOR ‘range’ from 0 to (length of ‘array’ - 1) DO

PRINT ‘array[range]’ followed by a space

END FOR

END

# FLOWCHART:



# Program for Checking whether two strings are Anagram or not

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the 1st input string:”

READ first string and CONVERT it to lowercase, STORE in ‘input1’

PRINT “Enter the 2nd input string”

READ second string and CONVERT it to lowercase, STORE in ‘input2’

IF length of ‘input1’ is not equal to length of ‘input2’ THEN

PRINT “Not an Anagram”

END IF

CONVERT ‘input1’ to a character array ‘array1’

CONVERT ‘input2’ to a character array ‘array2’

SORT ‘array1’

SORT ‘array2’

IF ‘array1’ is equal to ‘array2’ THEN

PRINT “Anagram”

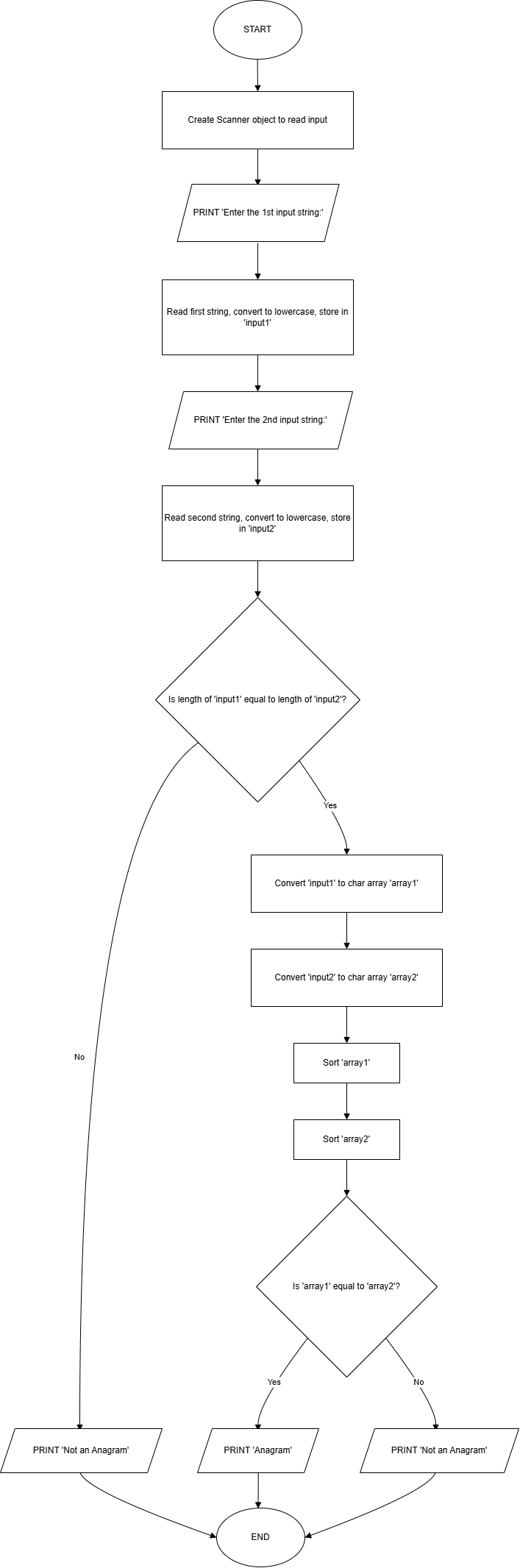
ELSE

PRINT “Not an Anagram”

End IF

END

# FLOWCHART:



# Program for validating the phone number using Pattern and Matcher classes in regex.

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the input string:”

READ input string and STORE in ‘input’

DEFINE a regex pattern:

Starts with either ‘+91’ or ‘0’

Followed by a digit from 6 to 9

Followed by 4 digits

Followed by a space

Followed by 5 digits

COMPILE the regex into ‘pattern’

CREATE a ‘matcher’ to match ‘input’ string against the ‘pattern’

IF ‘matcher’ matches THEN

PRINT “Valid Phone Number”

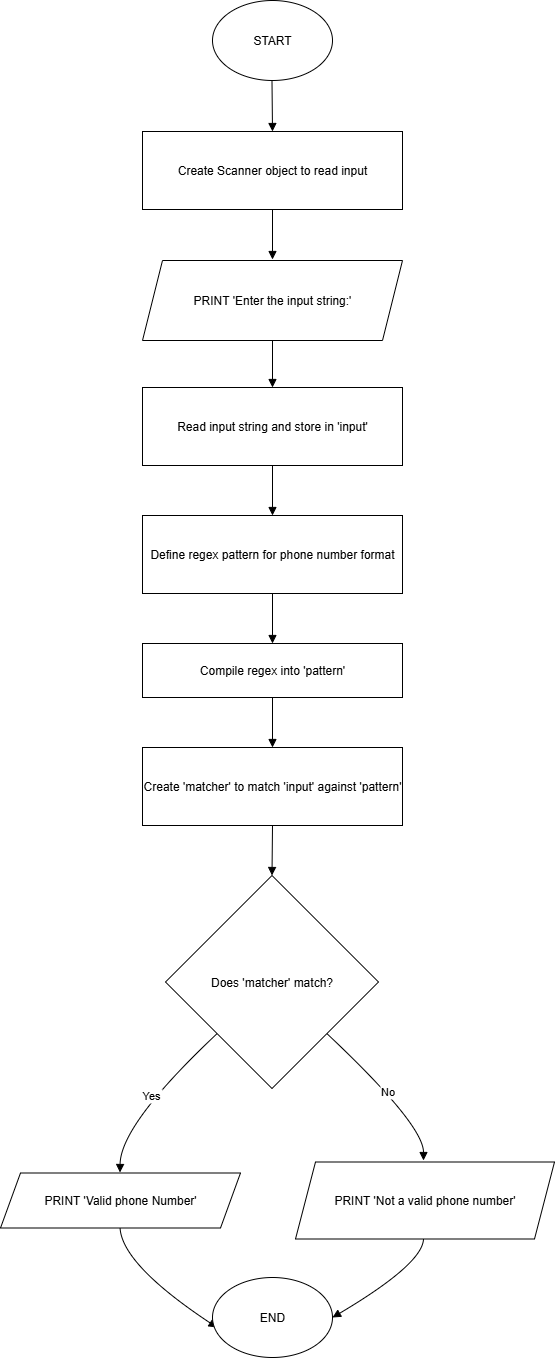
ELSE

PRINT “Not a valid phone number”

END IF

END

# FLOWCHART:



# Program for creating a thread and run that thread. Print the “Hi. Good morning!” for 5 times and with a time gap of 5 seconds.

# PESUDOCODE:

DEFINE a class ‘MyThread’ that EXTENDS ‘THREAD’

OVERRIDE the ‘run’ method

FOR ‘i’ from 0 to 4 DO

PRINT “Hi. Good morning...”

TRY

PAUSE the thread for 5000 milliseconds

CATCH

PRINT “InterruptedException is ocuurred…”

END TRY

END FOR

END CLASS

DEFINE a class ‘InterruptedException’

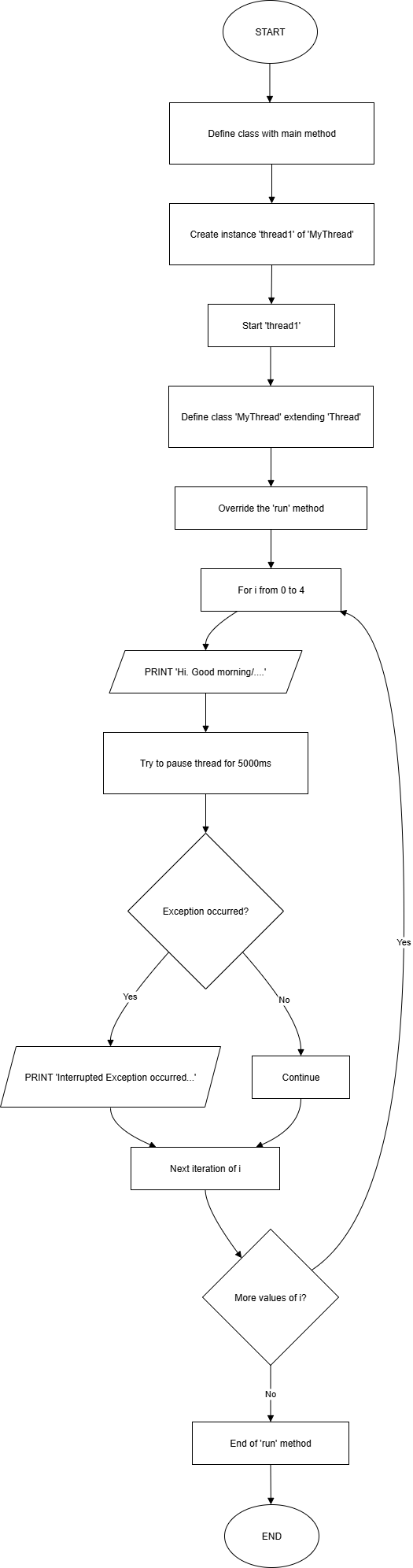
IN ‘main’ method:

CREATE an instance of ‘MyThread’ named ‘thread1’

START ‘thread1’ (this calls the ‘run’ method in a new thread)

END

# FLOW CHART:



# Program for printing Fibonacci Series up-to N

# PESUDOCODE:

BEGIN

CREATE a Scanner object to read input from user

PRINT “Enter the number of terms:”

READ value and STORE in ‘input’

INITIALIZE ‘a’ to 0 and ‘b’ to 1

PRINT “Fibonacci Series:”

FOR ‘i’ from 1 to ‘input’ DO

PRINT value of ‘a’ followed by a space

CALCULATE ‘c’ as ‘a + b’

ASSIGN ‘b’ to ‘a’

ASSIGN ‘c’ to ‘b’

END FOR

END

# FLOWCHART:

