1. Write an algorithm to find the sum of digits of a given number.

Example:

Input: 123

Output: 1 + 2 + 3 = 6

ALGORITHM:

1. Start

2. Initialize sum to 0.

3. Read the number: 123

4. Initialize a variable to store the number: number = 123

5. Repeat the following steps while the number is greater than 0:

a. Extract the last digit of the number: 123 % 10 = 3

b. Add the extracted digit to the sum: sum = sum + 3 = 0 + 3 = 3

c. Remove the last digit from the number: 123 / 10 = 12

6. Repeat:

a. Extract the last digit of the number: 12 % 10 = 2

b. Add the extracted digit to the sum: sum = sum + 2 = 3 + 2 = 5

c. Remove the last digit from the number: 12 / 10 = 1

7. Repeat:

a. Extract the last digit of the number: 1 % 10 = 1

b. Add the extracted digit to the sum: sum = sum + 1 = 5 + 1 = 6

c. Remove the last digit from the number: 1 / 10 = 0

8. The number is now 0, so exit the loop.

9. Display the original number: 123

10. Display the sum as a sum of individual digits: 1 + 2 + 3 = 6

11. End

PSEUDOCODE

1.Start

2. Initialize sum to 0.

3. Read the number.

4. Repeat the following steps while the number is greater than 0:

a. Extract the last digit of the number (number modulo 10).

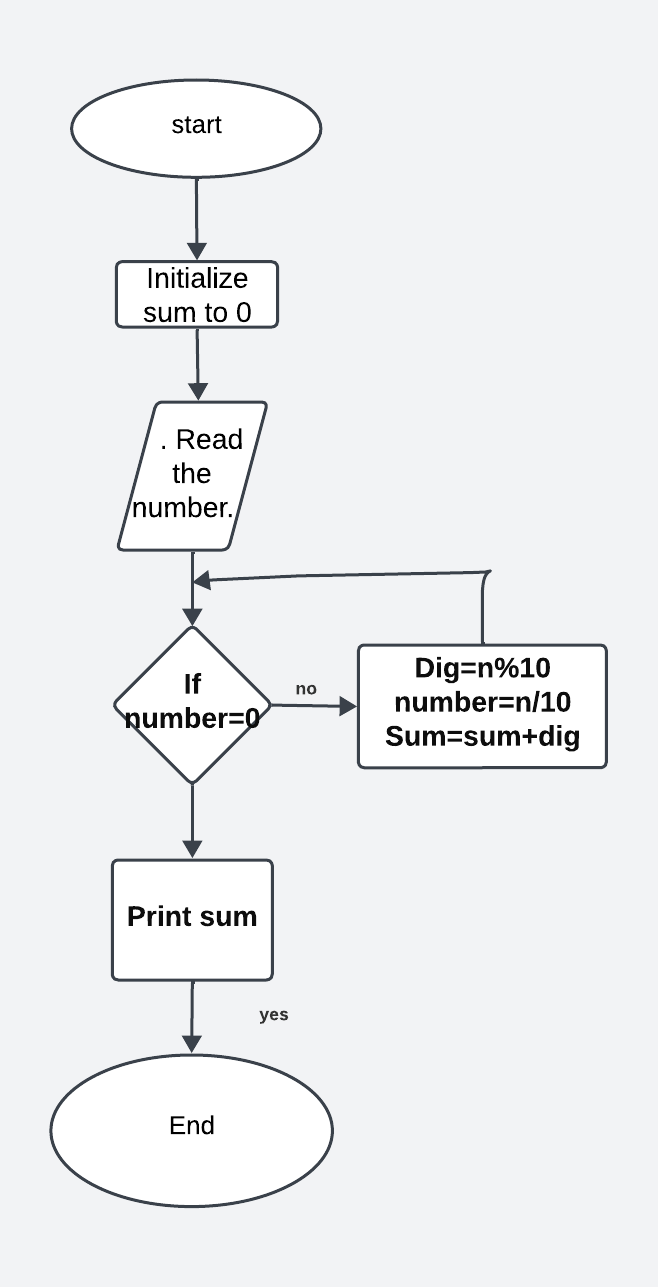
b. Add the extracted digit to the sum.

c. Remove the last digit from the number (number divided by 10).

5. Display the sum .

6. End

FLOWCHART:



**2.**Write an algorithm to calculate the factorial of a given number.

Example: Input: 5

Output: 5! = 5 \* 4 \* 3 \* 2 \* 1 = 120

ALGORITHM:

1.Start

2.Take a number as input.

3.Initialize a variable fact to 1.

4.If the number is 0 or 1, return 1.

5.Otherwise, loop from 1 to the input number:

a. Multiply fact by the loop variable.

6.Return fact.

7.End

PSEUDOCODE

1. Start

2. Read input number n

3. Set fact to 1

4. If n equals 0 or 1, then:

a. Return 1

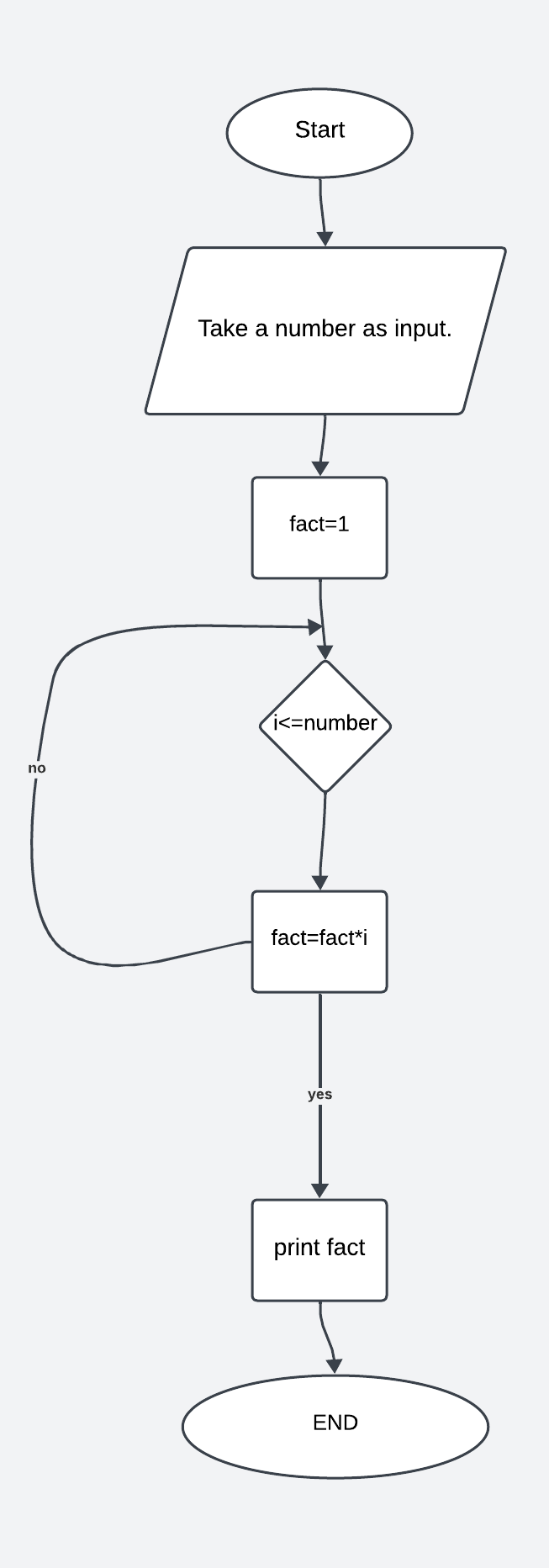
5. For i = 1 to n:

a. Multiply fact by i and update fact

6. print fact

7. End

FLOWCHART



3. Write an algorithm to check if a given string is a palindrome.

Example:

Input: "racecar"

Output: True

ALGORITHM:

1.Start

2.Read the input string.

3.Initialize two pointers, one at the beginning of the string (left pointer) and one at the end of the string (right pointer).

4.While the left pointer is less than the right pointer:

a. Compare the characters at the left and right pointers.

b. If they are not equal, the string is not a palindrome, return False.

c. Move the left pointer to the right and the right pointer to the left.

6.If the loop completes without finding any unequal characters, the string is a palindrome, return True.

7.End

PSEUDOCODE

1. Start

2. Read input string s

3. Set left pointer to the beginning of the string (0) and right pointer to the end of the string (length of s - 1)

4. While left pointer is less than right pointer:

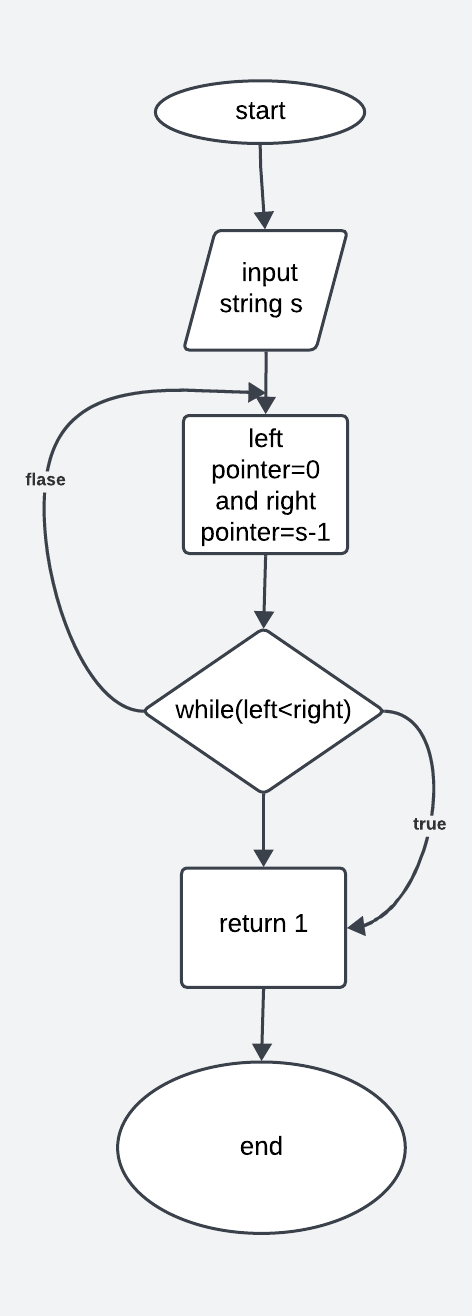
a. If s[left] is not equal to s[right], return 0 (indicating false)

b. Increment left pointer and decrement right point

5. Return 1

6. End

Flowchart:



6. Problem: Write an algorithm to generate the Fibonacci sequence up to a given number of terms.

Example:

Input: 7

Output: 0, 1, 1, 2, 3, 5, 8

ALGORITHM:

1.Start

2.Read the input number of terms.

3.Initialize variables to store the first two numbers of the Fibonacci sequence: a = 0 and b = 1 and sum=0.

4.Print a (the first number of the sequence).

5.Repeat the following steps for the remaining terms:

a. Print b (the current number of the sequence).

b. Calculate the next Fibonacci number: sum= a+ b.

c. Update a to b and b to sum.

6.print sum

7.End

PSEUDOCODE

1. Start

2. Read input number of terms n

3. Set a= 0 and b = 1and sum=0

4. Print a

5. Repeat n - 1 times

a. Print b

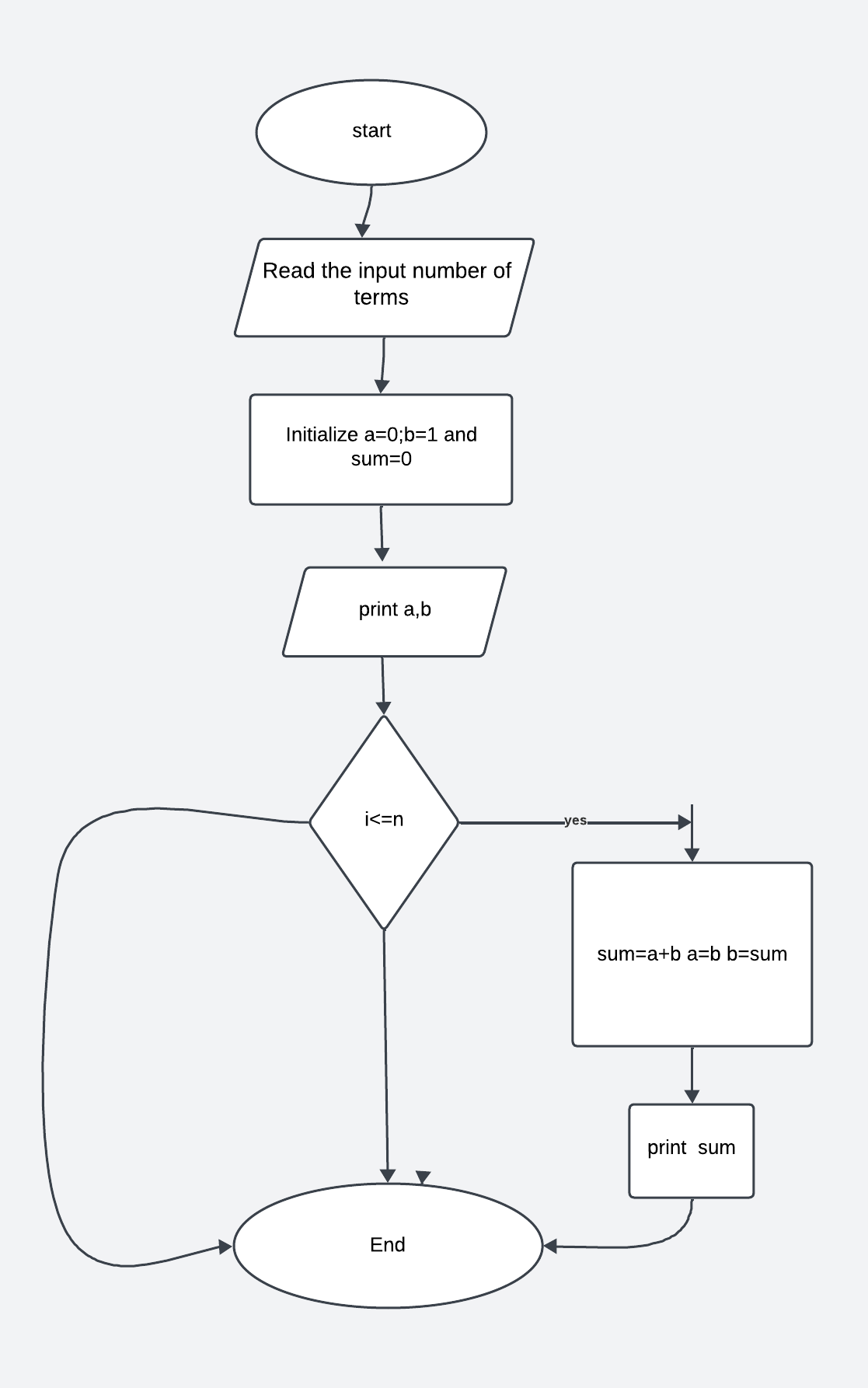
b. Calculate sum Fibonacci number: next = a + b

c. Update a to b and b to sum

6. print sum

7. End

Flowchart:



**9. Problem:** Write an algorithm to sort a list of numbers using the merge sort algorithm.

Example:

Input: [5, 3, 8, 1, 2]

Output: [1, 2, 3, 5, 8]

ALGORITHM:

1. start
2. Take a list of numbers as input.
3. If the list size is less than or equal to 1, return the list (it's already sorted).
4. Divide the list into two halves.
5. Recursively sort each half.
6. Merge the two sorted halves together.
7. Return the merged and sorted list.
8. end

PSEUDOCODE

FUNCTION mergesort(list)

IF LENGTH(list) <= 1 THEN

RETURN list

END IF

mid <- LENGTH(list) DIV 2

left <- mergeSort(list[0:mid])

right <- mergeSort(list[mid:LENGTH(list)])

RETURN merge(left, right)

END FUNCTION

FUNCTION merge(left, right)

result <- []

WHILE LENGTH(left) > 0 AND LENGTH(right) > 0 DO

IF left[0] <= right[0] THEN

result.APPEND(left.REMOVE(0))

ELSE

result.APPEND(right.REMOVE(0))

END IF

END WHILE

result += left + right

RETURN result

END FUNCTION

10.Write an algorithm to count the occurrences of each word in a given text.

Example: Input: "This is a sample text. This text contains sample words."

Output: {"This": 2, "is": 1, "a": 1, "sample": 2, "text": 2, "contains": 1, "words": 1}

ALGORITHM:

1.Start

2.Read the text.

3.Initialize an empty dictionary to store the word counts.

4.Split the text into words using whitespace as the delimiter.

5.For each word in the list of words:

a. If the word is already a key in the dictionary, increment its count by 1.

b. If the word is not in the dictionary, add it as a key with a count of 1.

6.Display the dictionary containing the word counts.

7.End

PSEUDOCODE

1. Start

2. Read text

3. Create an empty dictionary word\_count

4. Split text into words using whitespace as delimiter and store it in word\_list

5. For each word in word\_list:

a. If word exists as a key in word\_count: Increase its count by 1

b. Else:Add word as a key in word\_count with count 1

6. Display word\_count

7. End

FLOWCHART

