**ASSINGMENT 1**

**NAME BALAGUGAN**

**CLASS MECH AIML**

**COURSE PROGRAME SOLVING**

**CODE 21CSS101J**

**REG NO RA2411054040020**

**Write an algorithm, pseudocode, and flowchart for calculating the sum of all even numbers from 1 to n, where n is entered by the user.**

**Algorithm**

• Start.

• Prompt the user to enter a positive integer n.

• Initialize a variable sum to 0.

• Loop through all numbers from 1 to n.

• If the number is even (i.e., divisible by 2), add it to sum.

• After the loop ends, print the value of sum.

• End.

**Pseudocode**

START

PROMPT "Enter a positive integer (n): "

READ n

SET sum = 0

FOR i FROM 1 TO n DO

IF i MOD 2 = 0 THEN

sum = sum + i

ENDIF

ENDFOR

PRINT "The sum of all even numbers from 1 to", n, "is:", sum

END

**Flowchart**

• Start node.

• Input for n.

• Initialization of sum = 0.

• A loop iterating from 1 to n.

• A decision block to check if the current number is even (i % 2 == 0).

• Adding the number to sum if the condition is true.

• Exiting the loop and printing the result.

• End node.

**2. Factorial Using Recursion**

**Design a pseudocode, algorithm, and flowchart to calculate the factorial of a number using recursion.**

**Algorithm**

• Start.

• Define a recursive function factorial:

• If the input number n is 0 or 1, return 1 (base case).

• Otherwise, return n \* factorial(n-1) (recursive case).

• Prompt the user to input a positive integer n.

• Call the factorial function with the input value n.

• Display the result.

• End.

**Pseudocode**

FUNCTION factorial(n)

IF n = 0 OR n = 1 THEN

RETURN 1

ELSE

RETURN n \* factorial(n - 1)

ENDIF

END FUNCTION

START

PROMPT "Enter a positive integer: "

READ n

SET result = factorial(n)

PRINT "The factorial of", n, "is:", result

END

**Flowchart**

**• Start** node.

**• Input block** for the user to enter n.

• A rectangle to represent the recursive factorial function:

• A decision block checks if n=0n = 0n=0 or n=1n = 1n=1.

• If true, the function returns 1 (base case).

• Otherwise, the function recursively calls itself with n−1n - 1n−1 and multiplies the result by nnn (recursive case).

**• Output block** to display the result.

**• End** node.

**3. Prime Number Check**

**Write the algorithm, pseudocode, and flowchart to check whether a given number is a prime number or not.**

**Algorithm**

• Start.

• Prompt the user to enter a positive integer n.

• If n is less than or equal to 1, it is not a prime number.

• For numbers greater than 1:

• Initialize a loop to check divisors from 2 to n\sqrt{n}n​.

• If n is divisible by any number in this range, it is not a prime number.

• If no divisors are found, n is a prime number.

• Display the result.

• End.

**Pseudocode**

START

PROMPT "Enter a positive integer: "

READ n

IF n <= 1 THEN

PRINT n, "is not a prime number."

EXIT

ENDIF

SET isPrime = TRUE

FOR i FROM 2 TO √n DO

IF n MOD i = 0 THEN

SET isPrime = FALSE

BREAK

ENDIF

ENDFOR

IF isPrime THEN

PRINT n, "is a prime number."

ELSE

PRINT n, "is not a prime number."

ENDIF

END

**Flowchart**

The flowchart includes:

**• Start** node.

**• Input block** for n.

• A decision block to check if n≤1n \leq 1n≤1.

• If true, print "Not a prime number" and end.

• A loop to iterate from 2 to n\sqrt{n}n​:

• A decision block to check if nmod  i=0n \mod i = 0nmodi=0.

• If true, set isPrime = FALSE and exit the loop.

• After the loop:

• Print "Prime number" if isPrime is true.

• Otherwise, print "Not a prime number."

**• End** node.

**4. Finding the Largest Number in an Array**

**Create an algorithm, pseudocode, and flowchart for finding the largest number in an array of n elements provided by the user.**

**Algorithm**

• Start.

• Prompt the user to enter the size of the array n.

• Prompt the user to enter n elements of the array.

• Initialize a variable largest with the value of the first element of the array.

• Iterate through the array starting from the second element:

• If the current element is greater than largest, update largest to the current element.

• After the loop, largest contains the largest value in the array.

• Display the largest number.

• End.

**Pseudocode**

START

PROMPT "Enter the number of elements in the array (n): "

READ n

DECLARE array[n]

PROMPT "Enter the elements of the array: "

FOR i FROM 1 TO n DO

READ array[i]

ENDFOR

SET largest = array[1]

FOR i FROM 2 TO n DO

IF array[i] > largest THEN

SET largest = array[i]

ENDIF

ENDFOR

PRINT "The largest number in the array is:", largest

END

**Flowchart**

The flowchart includes:

**• Start** node.

**• Input block** for the number of elements n.

**• Input block** for the elements of the array.

• A rectangle to initialize largest with the first element of the array.

• A loop that iterates through the array from the second element to the last.

• A decision diamond checks if the current element is greater than largest.

• If true, a rectangle updates largest with the current element.

**• Output block** to display the largest number.

**• End** node

**5. Simple Interest Calculation**

**Write an algorithm, pseudocode, and flowchart to calculate simple interest based on the formula SI = (P × R × T) / 100, where P is the principal amount, R is the rate of interest, and T is the time in years.**

**Algorithm**

• Start.

• Prompt the user to enter the principal amount P.

• Prompt the user to enter the rate of interest R.

• Prompt the user to enter the time in years T.

• Calculate the simple interest using the formula: SI=P×R×T100\text{SI} = \frac{P \times R \times T}{100}SI=100P×R×T​

• Display the calculated simple interest SI.

• End.

**Pseudocode**

START

PROMPT "Enter the principal amount (P): "

READ P

PROMPT "Enter the rate of interest (R): "

READ R

PROMPT "Enter the time in years (T): "

READ T

SET SI = (P \* R \* T) / 100

PRINT "The Simple Interest (SI) is:", SI

END

**Flowchart**

The flowchart will include:

**• Start** node.

**• Input block** for the principal amount P.

**• Input block** for the rate of interest R.

**• Input block** for the time in years T.

• A rectangle to calculate the simple interest SI using the formula.

**• Output block** to display the value of SI.

**• End** node.

**6. Generate Fibonacci Series**

**Design the algorithm, pseudocode, and flowchart to generate the first n terms of the Fibonacci sequence**.

**Algorithm**

• Start.

• Prompt the user to enter the number of terms n to generate in the Fibonacci series.

• If n is less than or equal to 0, display an error message and stop.

• Initialize two variables:

• a = 0 (first term).

• b = 1 (second term).

• If n >= 1, display a.

• If n >= 2, display b.

• Use a loop to calculate and display the next terms of the Fibonacci sequence until n terms are generated:

• Compute next = a + b.

• Update a = b and b = next.

• Display next.

• End.

**Pseudocode**

START

PROMPT "Enter the number of terms (n): "

READ n

IF n <= 0 THEN

PRINT "Invalid input. Enter a positive integer."

EXIT

ENDIF

SET a = 0

SET b = 1

IF n >= 1 THEN

PRINT a

ENDIF

IF n >= 2 THEN

PRINT b

ENDIF

FOR i FROM 3 TO n DO

SET next = a + b

PRINT next

SET a = b

SET b = next

ENDFOR

END

**Flowchart**

**• Start** node.

**• Input block** for n.

• A decision diamond to check if n <= 0:

• If true, display an error message and go to **End**.

• Initialize a = 0 and b = 1.

• Decision blocks to check if n >= 1 and n >= 2, displaying a and b respectively.

• A loop to calculate and display the next Fibonacci terms:

• Compute next = a + b.

• Update a and b.

• Display next.

**• End** node.

**7. Number Guessing Game**

**Write an algorithm, pseudocode, and flowchart for a number guessing game where the program generates a random number between 1 and 100, and the user has to guess the number with hints provided (e.g., higher/lower).**

**Algorithm**

• Start.

• Generate a random number target between 1 and 100.

• Initialize a flag variable guessed = False.

• Repeat until guessed is True:

• Prompt the user to enter a guess.

• If the guess is equal to the target:

• Display "Congratulations! You guessed the number."

• Set guessed = True.

• If the guess is less than the target:

• Display "Too low, try again."

• If the guess is greater than the target:

• Display "Too high, try again."

• End.

**Pseudocode**

START

SET target = RANDOM(1, 100)

SET guessed = FALSE

WHILE guessed = FALSE DO

PROMPT "Enter your guess (1-100): "

READ guess

IF guess = target THEN

PRINT "Congratulations! You guessed the number."

SET guessed = TRUE

ELSE IF guess < target THEN

PRINT "Too low, try again."

ELSE

PRINT "Too high, try again."

ENDIF

ENDWHILE

END

**Flowchart**

**• Start** node.

• A rectangle to generate a random number (target).

• Initialization of the guessed flag to False.

• A loop that continues until guessed is True:

• A parallelogram for inputting the user's guess.

• A decision diamond to check if the guess equals the target:

• If true, display "Congratulations!" and exit the loop.

• If false, check if the guess is less than the target:

• Display "Too low, try again."

• Otherwise, display "Too high, try again."

**• End** node.

**8. Temperature Conversion**

**Develop an algorithm, pseudocode, and flowchart for converting a temperature value from Celsius to Fahrenheit using the formula F = (C × 9/5) + 32.**

**Algorithm**

• Start.

• Prompt the user to enter the temperature in Celsius (C).

• Calculate the temperature in Fahrenheit using the formula: F=(C×95)+32F = (C \times \frac{9}{5}) + 32F=(C×59​)+32

• Display the converted temperature in Fahrenheit (F).

• End.

**Pseudocode**

START

PROMPT "Enter the temperature in Celsius (C): "

READ C

SET F = (C \* 9 / 5) + 32

PRINT "The temperature in Fahrenheit (F) is:", F

END

**Flowchart**

**• Start** node.

**• Input block** for the user to enter the temperature in Celsius (C).

• A rectangle to calculate the Fahrenheit temperature using the formula F=(C×9/5)+32F = (C \times 9 / 5) + 32F=(C×9/5)+32.

**• Output block** to display the result FFF.

**• End** node.

**9. Count Vowels in a String**

**Write the algorithm, pseudocode, and flowchart to count the number of vowels in a given string input by the user.**

**Algorithm**

• Start.

• Prompt the user to enter a string.

• Initialize a counter variable vowelCount = 0.

• Loop through each character in the string:

• Convert the character to lowercase.

• If the character is a vowel ('a', 'e', 'i', 'o', 'u'), increment vowelCount by 1.

• Display the total number of vowels (vowelCount).

• End.

**Pseudocode**

START

PROMPT "Enter a string: "

READ inputString

SET vowelCount = 0

SET vowels = "aeiou"

FOR EACH character IN inputString DO

SET character = LOWERCASE(character)

IF character IN vowels THEN

SET vowelCount = vowelCount + 1

ENDIF

ENDFOR

PRINT "The number of vowels in the string is:", vowelCount

END

**Flowchart**

**• Start** node.

**• Input block** to read a string from the user.

• Initialize vowelCount to 0.

• A loop to iterate through each character of the string:

• Convert the character to lowercase.

• A decision diamond to check if the character is a vowel.

• If true, increment vowelCount.

• After the loop, an **output block** displays the total number of vowels.

**• End** node.

**10. Find the GCD of Two Numbers**

**Create an algorithm, pseudocode, and flowchart to find the greatest common divisor (GCD) of two numbers using the Euclidean algorithm.**

**Algorithm**

• Input: Two positive integers, a and b.

• Step 1: Divide a by b and find the remainder r, i.e., r = a % b.

• Step 2: If r == 0, then the GCD is b. Stop.

• Step 3: Otherwise, set a = b and b = r, and go back to Step 1.

• Output: The value of b when r == 0 is the GCD.

Pseudocode:

Function GCD(a, b):

While b != 0:

r = a % b

a = b

b = r

End While

Return a

End Function

**Flowchart**

**+---------------------+**

**| Start |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Input a, b |**

**+---------------------+**

**|**

**v**

**+---------------------+**

**| Is b == 0? |**

**+---------------------+**

**| |**

**Yes No**

**| |**

**v v**

**+---------------------+ +---------------------+**

**| Return a (GCD) | | r = a % b |**

**+---------------------+ +---------------------+**

**|**

**v**

**+---------------------+**

**| a = b, b = r |**

**+---------------------+**

**|**

**v**

**+-------------------+**

**| Repeat (Loop) |**

**+-------------------+**

 **Start**: The algorithm begins.

 **Input a, b**: The user inputs two integers, a and b.

 **Check if b == 0**: The algorithm checks if the second number is 0. If b is 0, the GCD is found and returned as a.

 **Calculate Remainder (r = a % b)**: If b is not 0, it calculates the remainder of a divided by b.

 **Update a and b**: Update a to b and b to r (the remainder). Then, the loop repeats the division process.

 **Repeat or End**: The process repeats until b becomes 0, at which point the loop ends and the result is returned as the GCD of the two numbers.