# WEEK - 1

## 1.Write a python program to check wheather a given number is even or odd.

### ALGORITHM:

1. Start
2. Read a number from the user
3. Check if the number is divisible by 2
4. If remainder is 0, print Even
5. Else, print Odd
6. Stop

In [2]:
```
n=int(input("enter a number"))
if n%2==0:
    print("number is even")
else:
    print("number is odd")
```

```
number is even
```

## 2. Write a Python program to check whether a number is positive, negative, or zero.¶

### ALGORITHM:

1. Start
2. Input a number n
3. If n > 0, print "Positive number"
4. Else if n < 0, print "Negative number"
5. Else, print "Zero"
6. Stop

In [3]:
```
n = int(input("enter a number:"))
if n>=0:
    print("positive number")
elif n<=0:
    print("negative number")
else:
    print("zero")
```

```
positive number
```

## 3. Write a Python program to find the largest among three numbers.

## ALGORITHM:

1. Start
2. Input three numbers a, b, and c
3. If a ≥ b and a ≥ c, then a is the largest
4. Else if b ≥ a and b ≥ c, then b is the largest
5. Else, c is the largest
6. Print the largest number
7. Stop

In [4]:
```
a = int(input("Enter first number: "))
b = int(input("Enter second number: "))
c = int(input("Enter third number: "))

if a >= b and a >= c:
    print("Largest number is:", a)
elif b >= a and b >= c:
    print("Largest number is:", b)
else:
    print("Largest number is:", c)
```

```
Largest number is: 7
```

## 4. Write a Python program to check whether a given number is a prime number.

## ALGORITHM:

1. Start
2. Input a number n
3. If n ≤ 1, then it is not a prime number
4. Else, repeat steps from i = 2 to n − 1
5. If n is divisible by i, then it is not a prime number
6. If no number divides n, then it is a prime number
7. Display the result
8. Stop

In [5]:
```
n = int(input("Enter a number: "))

if n <= 1:
    print("Not a prime number")
else:
    for i in range(2, n):
        if n % i == 0:
            print("Not a prime number")
            break
    else:
        print("Prime number")
```

```
Not a prime number
```

## WEEK - 2

### 5. Write a Python program to find the factorial of a number.

### ALGORITHM:

1. Start
2. Input a number n
3. If n < 0, factorial is not defined
4. Else, set fact = 1
5. Repeat from i = 1 to n
6. fact = fact × i
7. Print the value of fact
8. Stop

In [8]:
```
n = int(input("Enter a number: "))

if n < 0:
    print("Factorial is not defined for negative numbers")
else:
    fact = 1
    for i in range(1, n + 1):
        fact = fact * i
    print("Factorial of", n, "is:", fact)
```

```
Factorial of 5 is: 120
```

### 6. Write a Python program to check whether a number is a palindrome.

### ALGORITHM:

1. Start
2. Input a number n
3. Store the value of n in a temporary variable temp

4. Initialize rev = 0
5. While temp > 0
6. digit = temp % 10
7. rev = rev × 10 + digit
8. temp = temp // 10
9. If rev == n, then the number is a palindrome
10. Else, the number is not a palindrome
11. Display the result
12. Stop

In [9]:
```python
n = int(input("Enter a number: "))
temp = n
rev = 0

while temp > 0:
    digit = temp % 10
    rev = rev * 10 + digit
    temp = temp // 10

if rev == n:
    print("Palindrome number")
else:
    print("Not a palindrome number")
```

```
Palindrome number
```

## 7. Write a Python program to check whether a given string is a palindrome.

## ALGORITHM:

1. Start
2. Input a string s
3. Remove spaces (optional) and convert the string to lowercase
4. Reverse the string
5. If the original string is equal to the reversed string, then it is a palindrome
6. Else, it is not a palindrome
7. Display the result
8. Stop

In [10]:
```python
s = input("Enter a string: ")

# Convert to lowercase and remove spaces
s = s.replace(" ", "").lower()
```

```python
# Reverse the string
rev = s[::-1]

if s == rev:
    print("Palindrome string")
else:
    print("Not a palindrome string")
```

```
Palindrome string
```

## WEEK - 3

### 8. Write a Python program to print the Fibonacci series up to N terms.

### ALGORITHM:

1. Start
2. Input the number of terms n
3. Initialize first two terms: .a = 0 .b = 1
4. If n ≤ 0, print an error message
5. If n == 1, print a
6. Else, repeat from 1 to n
7. Print a .c = a + b .a = b .b = c
8. Stop

In [11]:
```python
n = int(input("Enter the number of terms: "))

a = 0
b = 1

if n <= 0:
    print("Please enter a positive number")
elif n == 1:
    print(a)
else:
    for i in range(n):
        print(a, end=" ")
        c = a + b
        a = b
        b = c
```

```
0 1 1 2 3
```

## 9. Write a Python program to find the sum of digits of a number.

## Algorithm

1. Start
2. Input a number n
3. Initialize sum = 0
4. While n > 0 .digit = n % 10 .sum = sum + digit .n = n // 10
5. Print the value of sum
6. Stop

In [12]:
```
n = int(input("Enter a number: "))
sum_digits = 0

while n > 0:
    digit = n % 10
    sum_digits = sum_digits + digit
    n = n // 10

print("Sum of digits:", sum_digits)
```

```
Sum of digits: 10
```

## 10. Write a Python program to count vowels and consonants in a string.

## ALGORITHM:

1. Start
2. Input a string s
3. Convert the string to lowercase
4. Initialize two counters: .vowels = 0 .consonants = 0
5. For each character in the string:
6. If the character is an alphabet:
7. If it is a vowel (a, e, i, o, u), increment vowels
8. Else, increment consonants
9. Display the number of vowels and consonants 10.Stop

In [15]:
```
s = input("Enter a string: ").lower()

vowels = 0
consonants = 0

for ch in s:
    if ch.isalpha():
        if ch in 'aeiou':
            vowels += 1
        else:
            consonants += 1

print("Number of vowels:", vowels)
print("Number of consonants:", consonants)
```

```
Number of vowels: 4
Number of consonants: 5
```

# WEEK - 4

## 11. Write a Python program to reverse a string without using built-in functions.

## ALGORITHM:

1. Start
2. Input a string s
3. Initialize an empty string rev = ""
4. For each character in s from the last to the first: .Add the character to rev
5. Print rev
6. Stop

In [16]:
```
s = input("Enter a string: ")
rev = ""

for i in range(len(s) - 1, -1, -1):
    rev += s[i]

print("Reversed string:", rev)
```

```
Reversed string: uoy evol i
```

## 12. Write a Python program to count the occurrence of each character in a string.

## ALGORITHM:

1. Start
2. Input a string s
3. Initialize an empty dictionary char_count = {}
4. For each character ch in the string: .If ch is already in char_count,

increment its value by 1 .Else, add ch to char_count with value 1

5. Print the dictionary char_count
6. Stop

In [18]:
```
s = input("Enter a string: ")
char_count = {}

for ch in s:
    if ch in char_count:
        char_count[ch] += 1
    else:
        char_count[ch] = 1

print("Character occurrences:")
for ch, count in char_count.items():
    print(f"{ch} : {count}")
```

```
Character occurrences:
M : 1
a : 1
n : 1
i : 1
```

## 13. Write a Python program to create a simple calculator using conditional statements

## ALGORITM:

1. Start
2. Input two numbers: num1 and num2
3. Input an operator (+, -, *, /)
4. Check the operator: .If operator is +, calculate num1 + num2 .Else if operator is -, calculate num1 - num2 .Else if operator is *, calculate num1 * num2 .Else if operator is /, check if num2 != 0 .If yes, calculate num1 / num2 .Else, print "Division by zero not allowed" .Else, print "Invalid operator"
5. Print the result
6. Stop

In [20]:
```
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
operator = input("Enter operator (+, -, *, /): ")

if operator == '+':
    result = num1 + num2
    print("Result:", result)
elif operator == '-':
    result = num1 - num2
    print("Result:", result)
elif operator == '*':
    result = num1 * num2
    print("Result:", result)
elif operator == '/':
    if num2 != 0:
        result = num1 / num2
        print("Result:", result)
    else:
        print("Division by zero is not allowed")
else:
    print("Invalid operator")
```

```
Result: 3.0
```

# WEEK - 5

## 14. Write a Python program to implement a menu-driven calculator using a loop (repeat until the user exits).

### ALGORITHM:

```
1. Start
2. Repeat until the user chooses to exit:
    A. Display the menu:
        a. Addition
        b. Subtraction
        c. Multiplication
        d. Division
        e. Exit
    B. Input the user’s choice
    C. If choice is 1, 2, 3, or 4: .Input two numbers num1 and
       num2 . Perform the corresponding operation . Display the
       result
    D. If choice is 5, exit the loop
    E. Else, print “Invalid choice”
3. Stop
```

In [21]:

```
while True:
    print("\nMenu:")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")
```

```python
    choice = input("Enter your choice (1-5): ")

    if choice == '5':
        print("Exiting the calculator. Goodbye!")
        break
    elif choice in ['1', '2', '3', '4']:
        num1 = float(input("Enter first number: "))
        num2 = float(input("Enter second number: "))

        if choice == '1':
            print("Result:", num1 + num2)
        elif choice == '2':
            print("Result:", num1 - num2)
        elif choice == '3':
            print("Result:", num1 * num2)
        elif choice == '4':
            if num2 != 0:
                print("Result:", num1 / num2)
            else:
                print("Division by zero is not allowed")
    else:
        print("Invalid choice, please try again.")
```

```
Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Result: 69.0

Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Invalid choice, please try again.

Menu:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Exiting the calculator. Goodbye!
```

## 15. Write a Python program to generate a multiplication table for a given number (loop until the user stops).

## ALGORITHM:

1.Start 2. Repeat until the user chooses to stop:

1. Input a number n for which the multiplication table is required
2. For i from 1 to 10: . Compute result = n × i . Print n × i = result
3. Ask the user if they want to continue (Yes/No) 4.If user chooses "No", exit the loop
4. Stop

In [23]:
```python
while True:
    num = int(input("Enter a number to generate its multiplication table: "))

    print(f"\nMultiplication Table for {num}:")
    for i in range(1, 11):
        print(f"{num} x {i} = {num * i}")

    choice = input("\nDo you want to generate another table? (yes/no): ").lowe
    if choice != "yes":
        print("Program stopped.")
        break
```

```
Multiplication Table for 6:
6 x 1 = 6
6 x 2 = 12
6 x 3 = 18
6 x 4 = 24
6 x 5 = 30
6 x 6 = 36
6 x 7 = 42
6 x 8 = 48
6 x 9 = 54
6 x 10 = 60
Program stopped.
```

## 16. Write a Python program to print different patterns using loop concepts (e.g., star patterns, number patterns).

## ALGORITHM:

1. Start
2. Input the number of rows n for the pattern
3. For each pattern type, use nested loops: . Outer loop: controls the rows

. Inner loop: controls the number of characters (* or numbers) in each row

4. Print the pattern row by row
5. Stop

In [22]:
```python
# Input number of rows
n = int(input("Enter the number of rows: "))

print("\n1. Star Pyramid Pattern")
for i in range(1, n + 1):
    # Print leading spaces
    for j in range(n - i):
        print(" ", end="")
    # Print stars
    for k in range(2 * i - 1):
        print("*", end="")
    print()

print("\n2. Right-angled Star Triangle")
for i in range(1, n + 1):
    for j in range(i):
        print("*", end="")
    print()

print("\n3. Number Pyramid Pattern")
num = 1
for i in range(1, n + 1):
    for j in range(1, i + 1):
        print(num, end=" ")
        num += 1
    print()
```

```
1. Star Pyramid Pattern
       *
      ***
     *****
    *******
   *********
  ***********
 *************
***************

2. Right-angled Star Triangle
*
**
***
****
*****
******
*******
********

3. Number Pyramid Pattern
1
2 3
4 5 6
7 8 9 10
11 12 13 14 15
16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31 32 33 34 35 36
```

# FUNCTION-BASED QUESTIONS

## WEEK - 6

### 17. Write a Python function that takes a user's name and prints a greeting message.

### ALGORITHM:

1. Start
2. Define a function named greet_user that takes one parameter name
3. Inside the function, print a greeting message using the name
4. Ask the user to enter their name
5. Call the function and pass the entered name as an argument
6. Stop

```
In [24]: def greet_user(name):
             print("Hello,", name + "! Welcome.")

         # Taking user input
         user_name = input("Enter your name: ")

         # Function call
         greet_user(user_name)
```

```
Hello, MANIKANTA! Welcome.
```

## 18. Write a Python function that accepts two numbers and returns their sum.

## ALGORITHM:

1. Start
2. Define a function named add_numbers that takes two parameters a and b
3. Inside the function, calculate the sum of a and b
4. Return the calculated sum
5. Ask the user to enter two numbers
6. Call the function with the two numbers as arguments
7. Display the returned result
8. Stop

```
In [25]: def add_numbers(a, b):
             return a + b

         # Taking user input
         num1 = float(input("Enter first number: "))
         num2 = float(input("Enter second number: "))

         # Function call
         result = add_numbers(num1, num2)

         # Displaying the result
         print("The sum is:", result)
```

```
The sum is: 6.0
```

# WEEK - 7

## 19. Write a Python recursive function to find the factorial of a number.

## ALGOROTHM:

1. Start
2. Define a recursive function named factorial that takes one parameter n
3. If n is 0 or 1, return 1 (base case)
4. Otherwise, return n * factorial(n - 1) (recursive case)
5. Ask the user to enter a number
6. Call the function with the given number
7. Display the result
8. Stop

In [26]:
```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Taking user input
num = int(input("Enter a number: "))

# Function call and output
print("Factorial of", num, "is:", factorial(num))
```

```
Factorial of 25 is: 15511210043330985984000000
```

## 20. Write a Python lambda function to check whether a number is even.

## ALGORITHM:

1. Start
2. Take a number as input from the user
3. Define a lambda function that checks if the number is divisible by 2
4. If the remainder is 0, the number is even
5. Display the result
6. Stop

In [27]:
```python
# Lambda function to check even number
is_even = lambda n: n % 2 == 0
```

```python
# Taking user input
num = int(input("Enter a number: "))

# Checking and displaying result
if is_even(num):
    print(num, "is an even number.")
else:
    print(num, "is an odd number.")
```

```
6 is an even number.
```

## 21. Write a Python program to calculate factorial using recursion with input validation.

## ALGORITHM:

1. Start
2. Define a recursive function factorial(n) . If n is 0 or 1, return 1 (base case) . Otherwise, return n * factorial(n - 1)
3. Ask the user to enter a number
4. Validate the input: . If the number is negative, display an error message .If the number is non-negative, proceed
5. Call the factorial function with the valid input
6. Display the result
7. Stop

In [29]:

```python
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)

# Taking user input
num = int(input("Enter a non-negative integer: "))

# Input validation
if num < 0:
    print("Error: Factorial is not defined for negative numbers.")
else:
    print("Factorial of", num, "is:", factorial(num))
```

```
Factorial of 5 is: 120
```

# PROJECT / ADVANCED QUESTIONS

## WEEK - 8

### 22. Write a Python program to create a Library Book Management System using functions

### ALGORITHM:

1. Start
2. Create an empty list to store books
3. Define a function add_book() to add a book to the library
4. Define a function remove_book() to remove a book from the library
5. Define a function display_books() to show all available books
6. Display a menu with choices: . Add Book . Remove Book . Display Books . Exit
7. Take the user’s choice
8. Call the corresponding function based on the choice
9. Repeat steps 6–8 until the user chooses to exit 10 .Stop

In [ ]:

In [31]:

```
# List to store books
library = []

def add_book():
    book = input("Enter book name to add: ")
    library.append(book)
    print("Book added successfully.")

def remove_book():
    book = input("Enter book name to remove: ")
    if book in library:
        library.remove(book)
        print("Book removed successfully.")
    else:
        print("Book not found.")

def display_books():
    if library:
        print("Books in the library:")
        for book in library:
            print("-", book)
    else:
        print("Library is empty.")
```

```python
# Main program
while True:
    print("\nLibrary Book Management System")
    print("1. Add Book")
    print("2. Remove Book")
    print("3. Display Books")
    print("4. Exit")

    choice = input("Enter your choice (1-4): ")

    if choice == "1":
        add_book()
    elif choice == "2":
        remove_book()
    elif choice == "3":
        display_books()
    elif choice == "4":
        print("Exiting program.")
        break
    else:
        print("Invalid choice. Please try again.")
```

```
Library Book Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Book not found.

Library Book Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Book added successfully.

Library Book Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Books in the library:
- 5

Library Book Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
```

```
Invalid choice. Please try again.

Library Book Management System
1. Add Book
2. Remove Book
3. Display Books
4. Exit
Exiting program.
```

# WEEK - 9

## 23. Write a Python project to build a Calculator using modular programming (separate module for operations).

## ALGORITHM:

### Main Program (calculator.py)

1. Start
2. Import the operations module
3. Display calculator menu (Add, Subtract, Multiply, Divide, Exit)
4. Take user choice
5. Ask the user to enter two numbers
6. Call the corresponding function from the operations module
7. Display the result
8. Repeat steps 3–7 until the user chooses Exit
9. Stop

### Operations Module (operations.py)

1. Start
2. Define function add(a, b) to return sum
3. Define function subtract(a, b) to return difference
4. Define function multiply(a, b) to return product D5. efine function divide(a, b) with division-by-zero check
5. Stop

In [1]:
```python
#operations.py
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b
```

```python
def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    else:
        return a / b


#main.py

while True:
    print("\nCalculator Menu")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == "5":
        print("Exiting Calculator.")
        break

    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == "1":
        print("Result:", add(num1, num2))
    elif choice == "2":
        print("Result:", subtract(num1, num2))
    elif choice == "3":
        print("Result:", multiply(num1, num2))
    elif choice == "4":
        print("Result:", divide(num1, num2))
    else:
        print("Invalid choice. Please try again.")
```

```
Calculator Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Result: 20.0

Calculator Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
```

```
Result: 66.0

Calculator Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Result: -20.0

Calculator Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Result: 27.0

Calculator Menu
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Exit
Exiting Calculator.
```

## WEEK - 10

### 24. Write a Python program that applies modular programming principles and defines multiple reusable functions.

### ALGORITHM:

1. Start
2. Define a function add(a, b) to return the sum
3. Define a function subtract(a, b) to return the difference
4. Define a function multiply(a, b) to return the product
5. Define a function divide(a, b) with division-by-zero check
6. Display a menu of operations
7. Take the user’s choice
8. Ask the user to enter two numbers
9. Call the appropriate function based on the user’s choice
10. Display the result
11. Repeat steps 6–10 until the user chooses to exit
12. Stop

In [49]:
```python
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b

# Main program
while True:
    print("\nModular Programming Calculator")
    print("1. Add")
    print("2. Subtract")
    print("3. Multiply")
    print("4. Divide")
    print("5. Exit")

    choice = input("Enter your choice (1-5): ")

    if choice == "5":
        print("Program exited.")
        break

    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == "1":
        print("Result:", add(num1, num2))
    elif choice == "2":
        print("Result:", subtract(num1, num2))
    elif choice == "3":
        print("Result:", multiply(num1, num2))
    elif choice == "4":
        print("Result:", divide(num1, num2))
    else:
        print("Invalid choice. Try again.")
```

```
Modular Programming Calculator
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
```

```
Result: 27.0

Modular Programming Calculator
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Result: 14.0

Modular Programming Calculator
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Result: 6.0

Modular Programming Calculator
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Result: 17.0

Modular Programming Calculator
1. Add
2. Subtract
3. Multiply
4. Divide
5. Exit
Program exited.
```

# WEEK - 11

## 25. Write a Python program using modular programming principles and demonstrate:

• Input validation • Testing (minimum 3 test cases) • Debugging practice with comments

## ALGORITHM:

1. Start
2. Define a function get_number() to take valid numeric input . Keep asking until the user enters a valid number
3. Define functions: . add(a, b) → returns sum . subtract(a, b) → returns difference

4. In the main program: . Call get_number() twice to get valid inputs . Call arithmetic functions and display results
5. Perform testing by calling functions with predefined values
6. Add debugging comments to explain possible errors and fixes
7. Stop

In [50]:
```
# Function for input validation
def get_number():
    while True:
        try:
            num = float(input("Enter a number: "))
            return num
        except ValueError:
            # Debugging: This error occurs if input is not a number
            print("Invalid input! Please enter a valid number.")

# Reusable function for addition
def add(a, b):
    return a + b

# Reusable function for subtraction
def subtract(a, b):
    return a - b

# Main program
print("Modular Programming Demo")

num1 = get_number()
num2 = get_number()

print("Addition Result:", add(num1, num2))
print("Subtraction Result:", subtract(num1, num2))

# -------------------------------
# TESTING SECTION
# -------------------------------
# Test Case 1
# Expected Output: 10
print("Test Case 1 (5 + 5):", add(5, 5))

# Test Case 2
# Expected Output: -2
print("Test Case 2 (3 - 5):", subtract(3, 5))

# Test Case 3
# Expected Output: 0
print("Test Case 3 (0 + 0):", add(0, 0))

# Debugging Note:
# If incorrect results appear, check:
# - Function logic
# - Input values passed to functions
# - Data types (int/float)
```

```
Modular Programming Demo
Addition Result: 70.0
Subtraction Result: 20.0
Test Case 1 (5 + 5): 10
Test Case 2 (3 - 5): -2
Test Case 3 (0 + 0): 0
```

# WEEK - 12

## 26. Write a Python project for a User Registration System with input validation, testing, and debugging documentation.

## ALGORITHM:

1. Start
2. Define functions: . get_username() → Take username input and validate (non-empty, alphanumeric) . get_password() → Take password input and validate (min 6 characters, contains a number) . register_user(username, password) → Store user info in a dictionary . display_users() → Display all registered users
3. In the main program: . Use a loop to allow multiple registrations . Ask user if they want to continue registering
4. Perform testing with predefined usernames and passwords
5. Add debugging comments to explain potential errors and fixes
6. Stop

In [52]:
```python
# Dictionary to store user info
users = {}

# Function for username input with validation
def get_username():
    while True:
        username = input("Enter username: ")
        if username.isalnum() and len(username) >= 3:
            return username
        else:
            # Debugging: Invalid usernames may cause registration failure
            print("Invalid username! Use at least 3 alphanumeric characters.")

# Function for password input with validation
def get_password():
    while True:
        password = input("Enter password: ")
        if len(password) >= 6 and any(char.isdigit() for char in password):
            return password
        else:
            # Debugging: Weak passwords may compromise security
            print("Invalid password! Must be at least 6 characters with a numb

# Function to register user
def register_user(username, password):
    if username in users:
        print("Username already exists!")
    else:
        users[username] = password
        print("User registered successfully.")

# Function to display all users
def display_users():
    if users:
        print("Registered Users:")
        for user in users:
            print("-", user)
    else:
        print("No users registered yet.")

# Main program
print("=== User Registration System ===")

while True:
    uname = get_username()
    pwd = get_password()
    register_user(uname, pwd)

    choice = input("Register another user? (yes/no): ").lower()
    if choice != "yes":
        break

display_users()

# --------------------------------
# TESTING SECTION
# --------------------------------
print("\n=== TEST CASES ===")
# Test Case 1: Valid input
register_user("Alice123", "pass123")  # Expected: User registered successfully
# Test Case 2: Duplicate username
register_user("Alice123", "newpass456")  # Expected: Username already exists!
# Test Case 3: Weak password check
# Should fail input validation and ask again manually

# Debugging Notes:
# 1. Invalid username: Check if isalnum() fails or length < 3
# 2. Weak password: Ensure it contains at least one number and min length 6
# 3. Duplicate username: Check dictionary keys
```

```
=== User Registration System ===
Invalid password! Must be at least 6 characters with a number.
User registered successfully.
Registered Users:
- mani143

=== TEST CASES ===
User registered successfully.
Username already exists!
```

```
In [ ]:
```