1.Implement factorial using recursion

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n-1)

factorial(5)

120

2.Program using doubly linked list

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.next = None

self.prev = None

class DoublyLinkedList:

def \_\_init\_\_(self):

self.head = None

def append(self, data):

new\_node = Node(data)

if self.head is None:

self.head = new\_node

else:

curr\_node = self.head

while curr\_node.next is not None:

curr\_node = curr\_node.next

curr\_node.next = new\_node

new\_node.prev = curr\_node

def prepend(self, data):

new\_node = Node(data)

if self.head is None:

self.head = new\_node

else:

new\_node.next = self.head

self.head.prev = new\_node

self.head = new\_node

def delete(self, data):

curr\_node = self.head

while curr\_node is not None:

if curr\_node.data == data:

if curr\_node.prev is not None:

curr\_node.prev.next = curr\_node.next

else:

self.head = curr\_node.next

if curr\_node.next is not None:

curr\_node.next.prev = curr\_node.prev

break

curr\_node = curr\_node.next

def print\_list(self):

curr\_node = self.head

while curr\_node is not None:

print(curr\_node.data)

curr\_node = curr\_node.next

d = DoublyLinkedList()

d.append(1)

d.append(2)

d.append(3)

d.print\_list()

1

2

3

d.prepend(0)

d.print\_list()

0

1

2

3

d.delete(2)

d.print\_list()

0

1

3

3.Implement multiple inheritance using interface

class Printable:

def print(self):

pass

class Serializable:

def serialize(self):

pass

class MyClass(Printable, Serializable):

def \_\_init\_\_(self, data):

self.data = data

def print(self):

print(self.data)

def serialize(self):

return str(self.data)

obj = MyClass(42)

obj.print() # Output: 42

serialized = obj.serialize()

print(serialized) # Output: '42'

4.Print all pronic numbers between 1 and 100

for i in range(1, 101):

for j in range(1, i):

if i == j \* (j + 1):

print(i)

break

output

2

6

12

20

30

42

56

72

90

5.Implement method overloading & overiding in python

class Shape:

def area(self):

pass

class Rectangle(Shape):

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

def area(self):

return self.width \* self.height

class Circle(Shape):

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14 \* self.radius \* self.radius

overriding

class Shape:

def area(self):

pass

class Rectangle(Shape):

def \_\_init\_\_(self, width, height):

self.width = width

self.height = height

def area(self):

return self.width \* self.height

class Circle(Shape):

def \_\_init\_\_(self, radius):

self.radius = radius

def area(self):

return 3.14 \* self.radius \* self.radius

6.Program to find duplicate values for ArrayList

my\_list = [1, 2, 3, 2, 4, 3, 5, 6, 5]

duplicates = []

for value in my\_list:

if my\_list.count(value) > 1 and value not in duplicates:

duplicates.append(value)

print("Duplicate values in the list:", duplicates)

output:

Duplicate values in the list: [2, 3, 5]

7.Python program to print the elements of an array in reverse order

my\_array = [1, 2, 3, 4, 5]

# Using a for loop to iterate over the elements in reverse order

for i in range(len(my\_array)-1, -1, -1):

print(my\_array[i], end=" ")

output

5 4 3 2 1

8. Python program to determine whether the given number is a Harshad Number

num = int(input("Enter a number: "))

sum\_of\_digits = 0

for digit in str(num):

sum\_of\_digits += int(digit)

if num % sum\_of\_digits == 0:

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

9. Implement a program to merge two Arrays

# Sample arrays

array1 = [1, 2, 3]

array2 = [4, 5, 6]

# Merge the arrays using the + operator

merged\_array = array1 + array2

# Print the merged array

print("Merged array:", merged\_array)

# Sample arrays

array1 = [1, 2, 3]

array2 = [4, 5, 6]

# Merge the arrays using the extend() method

array1.extend(array2)

# Print the merged array

print("Merged array:", array1)

10. Program to find duplicate values for ArrayList

import java.util.ArrayList;

import java.util.HashSet;

public class FindDuplicates {

public static void main(String[] args) {

// Create an ArrayList with duplicate values

ArrayList<Integer> numbers = new ArrayList<>();

numbers.add(1);

numbers.add(2);

numbers.add(3);

numbers.add(2);

numbers.add(4);

numbers.add(1);

// Create a HashSet to store the unique elements

HashSet<Integer> seen = new HashSet<>();

// Iterate over the ArrayList and check for duplicates

for (Integer number : numbers) {

if (!seen.add(number)) {

System.out.println("Duplicate value found: " + number);

}

}

}

}

Output

Duplicate value found: 2

Duplicate value found: 1

11.Implement a program to sort a map by value / Key

Sorting a Map by value

import java.util.\*;

public class SortMapByValue {

public static void main(String[] args) {

// Create a HashMap with some key-value pairs

HashMap<String, Integer> map = new HashMap<>();

map.put("Alice", 22);

map.put("Bob", 18);

map.put("Charlie", 24);

map.put("Dave", 20);

// Create a list from the HashMap entries

List<Map.Entry<String, Integer>> list = new ArrayList<>(map.entrySet());

// Sort the list by value using a custom Comparator

Collections.sort(list, new Comparator<Map.Entry<String, Integer>>() {

@Override

public int compare(Map.Entry<String, Integer> o1, Map.Entry<String, Integer> o2) {

return o1.getValue().compareTo(o2.getValue());

}

});

// Create a new LinkedHashMap to store the sorted entries

LinkedHashMap<String, Integer> sortedMap = new LinkedHashMap<>();

for (Map.Entry<String, Integer> entry : list) {

sortedMap.put(entry.getKey(), entry.getValue());

}

// Print the sorted map

System.out.println("Sorted Map by Value: " + sortedMap);

}

}

Sorting a Map by Key

import java.util.\*;

public class SortMapByKey {

public static void main(String[] args) {

// Create a HashMap with some key-value pairs

HashMap<String, Integer> map = new HashMap<>();

map.put("Alice", 22);

map.put("Bob", 18);

map.put("Charlie", 24);

map.put("Dave", 20);

// Create a TreeMap to store the sorted entries by key

TreeMap<String, Integer> sortedMap = new TreeMap<>(map);

// Print the sorted map

System.out.println("Sorted Map by Key: " + sortedMap);

}

}

12.Write a python Program for Fibonacci series.

# Get the number of terms for the Fibonacci sequence

n = int(input("Enter the number of terms for the Fibonacci sequence: "))

# Initialize the first two terms of the sequence

a, b = 0, 1

# Print the first n terms of the sequence

for i in range(n):

print(a, end=" ")

# Calculate the next term of the sequence

c = a + b

# Update a and b for the next iteration

a, b = b, c

output

Enter the number of terms for the Fibonacci sequence: 10

0 1 1 2 3 5 8 13 21 34

13.Python program to print the elements of an array in reverse order

# Initialize an array

arr = [1, 2, 3, 4, 5]

# Print the array in reverse order

for i in range(len(arr)-1, -1, -1):

print(arr[i], end=" ")

output

5 4 3 2 1

14. Write a python Program for Fibonacci series.

# Get the number of terms for the Fibonacci sequence

n = int(input("Enter the number of terms for the Fibonacci sequence: "))

# Initialize the first two terms of the sequence

a, b = 0, 1

# Print the first n terms of the sequence

for i in range(n):

print(a, end=" ")

# Calculate the next term of the sequence

c = a + b

# Update a and b for the next iteration

a, b = b, c

Output

Enter the number of terms for the Fibonacci sequence: 10

0 1 1 2 3 5 8 13 21 34

15. Constructor Overloading

class Person:

def \_\_init\_\_(self, name=None, age=None):

self.name = name

self.age = age

def display(self):

print("Name:", self.name)

print("Age:", self.age)

# Create a person with name and age

p1 = Person("Alice", 25)

p1.display()

# Create a person with only name

p2 = Person("Bob")

p2.display()

# Create a person with only age

p3 = Person(age=30)

p3.display()

# Create a person with no arguments

p4 = Person()

p4.display()

Output

Name: Alice

Age: 25

Name: Bob

Age: None

Name: None

Age: 30

Name: None

Age: None

16. Python program to print the elements of an array in reverse order

# Define an array of integers

arr = [1, 2, 3, 4, 5]

# Print the array in reverse order

for i in range(len(arr)-1, -1, -1):

print(arr[i], end=" ")

output

5 4 3 2 1

17.Implement Exception Handling without Catch block

# Define a function that raises an exception

def divide\_by\_zero(x, y):

if y == 0:

raise Exception("Cannot divide by zero")

return x / y

# Call the function inside a try block

try:

result = divide\_by\_zero(10, 0)

except Exception as e:

print("An exception occurred:", e)

else:

print("Result:", result)

finally:

print("Execution complete")

output

An exception occurred: Cannot divide by zero

Execution complete

18Python program to determine whether the given number is a Harshad Number

# Function to check if a number is a Harshad number

def is\_harshad(num):

# Find the sum of digits of the number

sum\_of\_digits = sum([int(digit) for digit in str(num)])

# Check if the number is divisible by the sum of its digits

return num % sum\_of\_digits == 0

# Test the function with some examples

print(is\_harshad(18)) # True

print(is\_harshad(19)) # False

Output

True

False

19. Compare StringBuffer with a string

In Java, **StringBuffer** and **String** are two classes that are used for working with text. However, they have some key differences in terms of how they handle strings.

A **StringBuffer** is a mutable sequence of characters. This means that the contents of a **StringBuffer** can be modified after it has been created. In contrast, a **String** is immutable. Once a **String** object is created, its contents cannot be changed.

Here are some of the key differences between **StringBuffer** and **String**:

1. Mutability: As mentioned above, **StringBuffer** is mutable, while **String** is immutable.
2. Thread-safety: **StringBuffer** is thread-safe, which means that it can be safely accessed and modified by multiple threads at the same time without causing any issues. **String**, on the other hand, is not thread-safe.
3. Performance: Because **StringBuffer** is mutable, it can be more efficient than **String** in certain situations where many modifications are being made to a string. This is because **StringBuffer** avoids the need to create new string objects every time a modification is made.
4. API: The **String** class has a more extensive API than **StringBuffer**, with many built-in methods for working with strings, such as **startsWith()**, **endsWith()**, **indexOf()**, and **substring()**. **StringBuffer** has fewer built-in methods, but they are focused on manipulating the contents of the string.

In summary, **StringBuffer** is a mutable, thread-safe sequence of characters that can be more efficient than **String** in certain situations where many modifications are being made to a string. However, **String** has a more extensive API and is generally preferred for situations where immutability and thread-safety are not concerns.

20.Python program to print the elements of an array in reverse order

# Define an array of integers

arr = [1, 2, 3, 4, 5]

# Use a for loop to iterate over the array in reverse order

for i in range(len(arr)-1, -1, -1):

# Print each element of the array

print(arr[i])

Output

5

4

3

2

1

21. remove duplicates from sorted array

def removeDuplicates(nums):

"""

:type nums: List[int]

:rtype: int

"""

if len(nums) == 0:

return 0

i = 0

for j in range(1, len(nums)):

if nums[j] != nums[i]:

i += 1

nums[i] = nums[j]

return i + 1

22. Python program to print the elements of an array in reverse order

# Define an array

arr = [1, 2, 3, 4, 5]

# Reverse the array using slicing

arr\_reverse = arr[::-1]

# Print the reversed array

print(arr\_reverse)

Output

[5, 4, 3, 2, 1]

23. swap two numbers without using temporary variable

# Define two numbers

x = 5

y = 10

# Print the original values

print("Before swap: x =", x, ", y =", y)

# Swap the values without a temporary variable

x = x + y

y = x - y

x = x - y

# Print the swapped values

print("After swap: x =", x, ", y =", y)

output

Before swap: x = 5 , y = 10

After swap: x = 10 , y = 5

24.Python program to create a doubly linked list from a ternary tree

class Node:

def \_\_init\_\_(self, val):

self.val = val

self.left = None

self.middle = None

self.right = None

self.prev = None

self.next = None

class DoublyLinkedList:

def \_\_init\_\_(self):

self.head = None

self.tail = None

def add\_node(self, val):

new\_node = Node(val)

if self.head is None:

self.head = self.tail = new\_node

else:

self.tail.next = new\_node

new\_node.prev = self.tail

self.tail = new\_node

def traverse\_tree(root, dll):

if root is None:

return

traverse\_tree(root.left, dll)

dll.add\_node(root.val)

traverse\_tree(root.middle, dll)

traverse\_tree(root.right, dll)

def create\_dll\_from\_ternary\_tree(root):

dll = DoublyLinkedList()

traverse\_tree(root, dll)

return dll

25.Find Maximum repeated charcter count in a string

def max\_repeated\_char\_count(string):

char\_count = {}

for char in string:

if char in char\_count:

char\_count[char] += 1

else:

char\_count[char] = 1

max\_count = max(char\_count.values())

return max\_count

max\_repeated\_char\_count("hello world")

2

max\_repeated\_char\_count("abccdefg")

2

max\_repeated\_char\_count("aaaabbbbcccdde")

4

26. Implement quick sorting

def quick\_sort(arr):

if len(arr) <= 1:

return arr

pivot = arr[len(arr) // 2]

left = [x for x in arr if x < pivot]

middle = [x for x in arr if x == pivot]

right = [x for x in arr if x > pivot]

return quick\_sort(left) + middle + quick\_sort(right)

arr = [3, 7, 1, 9, 2, 5, 6]

quick\_sort(arr)

[1, 2, 3, 5, 6, 7, 9]

27.Find duplicate elements in a string

def find\_duplicates(string):

duplicates = []

for char in string:

if string.count(char) > 1 and char not in duplicates:

duplicates.append(char)

return duplicates

find\_duplicates("hello")

['l']

find\_duplicates("python programming")

['o', 'r', 'm', 'g', 'n']

find\_duplicates("aabbccdd")

['a', 'b', 'c', 'd']

28.Python program to determine whether the given number is a Harshad Number

def is\_harshad(number):

# Convert number to string and calculate the sum of its digits

digit\_sum = sum(int(digit) for digit in str(number))

# Check if the number is divisible by the sum of its digits

return number % digit\_sum == 0

is\_harshad(18)

True

is\_harshad(19)

False

is\_harshad(1729)

True

29. Programs with list and tuples

def find\_largest(numbers):

largest = numbers[0]

for number in numbers:

if number > largest:

largest = number

return largest

numbers = [5, 10, 2, 8, 3]

print(find\_largest(numbers))

def tuple\_sum(numbers):

total = 0

for number in numbers:

total += number

return total

numbers = (5, 10, 2, 8, 3)

print(tuple\_sum(numbers))

def remove\_duplicates(numbers):

unique\_numbers = []

for number in numbers:

if number not in unique\_numbers:

unique\_numbers.append(number)

return unique\_numbers

numbers = [5, 10, 2, 8, 3, 5, 8, 10]

print(remove\_duplicates(numbers))

def concatenate\_lists(list1, list2):

return list1 + list2

list1 = [1, 2, 3]

list2 = [4, 5, 6]

print(concatenate\_lists(list1, list2))

30. Implement dictionary

# create an empty dictionary

my\_dict = {}

# add key-value pairs

my\_dict["apple"] = 1

my\_dict["banana"] = 2

my\_dict["orange"] = 3

# access values by key

print(my\_dict["apple"]) # prints 1

print(my\_dict["banana"]) # prints 2

print(my\_dict["orange"]) # prints 3

# iterate over keys

for key in my\_dict:

print(key) # prints "apple", "banana", "orange"

# iterate over values

for value in my\_dict.values():

print(value) # prints 1, 2, 3

# iterate over key-value pairs

for key, value in my\_dict.items():

print(key, value) # prints "apple 1", "banana 2", "orange 3"

31. Python program to determine whether the given number is a Harshad Number

def is\_harshad\_number(num):

"""

Check whether the given number is a Harshad Number or not

"""

# convert the number to a string

num\_str = str(num)

# calculate the sum of the digits

digit\_sum = sum(int(digit) for digit in num\_str)

# check if the number is divisible by the sum of its digits

if num % digit\_sum == 0:

return True

else:

return False

# example usage

num = 18

if is\_harshad\_number(num):

print(num, "is a Harshad Number")

else:

print(num, "is not a Harshad Number")

32. Python program to create a doubly linked list from a ternary tree

class Node:

"""

A class to represent a node in a ternary tree

"""

def \_\_init\_\_(self, value):

self.value = value

self.left = None

self.mid = None

self.right = None

class DoublyLinkedListNode:

"""

A class to represent a node in a doubly linked list

"""

def \_\_init\_\_(self, value):

self.value = value

self.prev = None

self.next = None

class DoublyLinkedList:

"""

A class to represent a doubly linked list

"""

def \_\_init\_\_(self):

self.head = None

self.tail = None

def add\_node(self, value):

"""

Adds a node to the end of the linked list

"""

new\_node = DoublyLinkedListNode(value)

if self.tail:

self.tail.next = new\_node

new\_node.prev = self.tail

self.tail = new\_node

else:

self.head = new\_node

self.tail = new\_node

def ternary\_tree\_to\_doubly\_linked\_list(root):

"""

Converts a ternary tree to a doubly linked list

"""

linked\_list = DoublyLinkedList()

stack = [root]

while stack:

node = stack.pop()

linked\_list.add\_node(node.value)

if node.right:

stack.append(node.right)

if node.mid:

stack.append(node.mid)

if node.left:

stack.append(node.left)

return linked\_list

# create a ternary tree

root = Node(1)

root.left = Node(2)

root.mid = Node(3)

root.right = Node(4)

root.left.left = Node(5)

root.left.mid = Node(6)

root.left.right = Node(7)

root.mid.left = Node(8)

root.mid.mid = Node(9)

root.mid.right = Node(10)

root.right.left = Node(11)

root.right.mid = Node(12)

root.right.right = Node(13)

# convert the ternary tree to a doubly linked list

linked\_list = ternary\_tree\_to\_doubly\_linked\_list(root)

# print the linked list

node = linked\_list.head

while node:

print(node.value)

node = node.next

Output

1

2

5

6

7

3

8

9

10

4

11

12

13

33. compare two arrays and return the common elements

def common\_elements(arr1, arr2):

"""

This function takes two arrays as input and returns a new array

that contains the common elements between them.

"""

common = []

for i in arr1:

if i in arr2 and i not in common:

common.append(i)

return common

# Example usage

arr1 = [1, 2, 3, 4, 5, 6]

arr2 = [4, 5, 6, 7, 8, 9]

result = common\_elements(arr1, arr2)

print(result) # Output: [4, 5, 6]

34. Write a python Program to find whether a string or number is palindrome or not.

def is\_palindrome(s):

"""

This function takes a string or number as input and returns True

if it is palindrome, False otherwise.

"""

s = str(s) # Convert number to string if input is a number

n = len(s)

for i in range(n//2):

if s[i] != s[n-i-1]:

return False

return True

# Example usage

s1 = "racecar"

s2 = 12321

s3 = "hello"

print(is\_palindrome(s1)) # Output: True

print(is\_palindrome(s2)) # Output: True

print(is\_palindrome(s3)) # Output: False

35. Implement more than one interface in a single class

from abc import ABC, abstractmethod

class Shape(ABC):

@abstractmethod

def area(self):

pass

@abstractmethod

def perimeter(self):

pass

class Polygon(ABC):

@abstractmethod

def sides(self):

pass

class Rectangle(Shape, Polygon):

def \_\_init\_\_(self, length, width):

self.length = length

self.width = width

def area(self):

return self.length \* self.width

def perimeter(self):

return 2 \* (self.length + self.width)

def sides(self):

return 4

36. Python program to determine whether the given number is a Harshad Number

def is\_harshad(num):

# Compute the sum of digits

digit\_sum = sum(int(digit) for digit in str(num))

# Check if the number is divisible by the sum of its digits

return num % digit\_sum == 0

# Example usage

num = 18

if is\_harshad(num):

print(f"{num} is a Harshad number")

else:

print(f"{num} is not a Harshad number")

37. Implement a program for encapsulation

class BankAccount:

def \_\_init\_\_(self, account\_number, balance):

self.\_\_account\_number = account\_number

self.\_\_balance = balance

def deposit(self, amount):

self.\_\_balance += amount

def withdraw(self, amount):

if self.\_\_balance >= amount:

self.\_\_balance -= amount

else:

print("Insufficient balance")

def get\_balance(self):

return self.\_\_balance

account = BankAccount("1234567890", 1000)

account.deposit(500)

account.withdraw(2000)

print(account.get\_balance())

38. Print all pronic numbers between 1 and 100

for i in range(1, 10):

pronic = i \* (i+1)

if pronic > 100:

break

print(pronic, end=' ')

Output

2 6 12 20 30 42 56 72 90

39. convert string to char and vice versa

string = "Hello, World!"

# Accessing individual characters

print(string[0]) # Output: 'H'

print(string[7]) # Output: 'W'

char = 'A'

# Converting a character to a string

string = str(char)

print(string) # Output: 'A'

string = "Hello, World!"

# Converting a string to a list of characters

char\_list = list(string)

print(char\_list) # Output: ['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd', '!']

40. Iterate the LinkedHashMap values

from collections import OrderedDict

# Create an OrderedDict

od = OrderedDict()

od['one'] = 1

od['two'] = 2

od['three'] = 3

# Iterate over the values of the OrderedDict

for value in od.values():

print(value)

Output

1

2

3

41. Implement a program for abstraction

from abc import ABC, abstractmethod

class Animal(ABC):

@abstractmethod

def sound(self):

pass

class Dog(Animal):

def sound(self):

print("Woof!")

d = Dog()

d.sound() # Output: Woof!

42. Print all pronic numbers between 1 and 100

for i in range(1, 101):

for j in range(1, i):

if i == j \* (j + 1):

print(i)

break

43. Implement a program to handle more than one exception

try:

a = int(input("Enter a number: "))

b = int(input("Enter another number: "))

result = a / b

print("The result is:", result)

except ZeroDivisionError:

print("Error: Cannot divide by zero")

except ValueError:

print("Error: Invalid input")

44. Python program to create a doubly linked list from a ternary tree

class Node:

def \_\_init\_\_(self, data):

self.data = data

self.left = None

self.right = None

self.middle = None

class DLLNode:

def \_\_init\_\_(self, data):

self.data = data

self.prev = None

self.next = None

class TernaryTree:

def \_\_init\_\_(self, root):

self.root = root

def traverse\_tree(self, node):

if node:

self.traverse\_tree(node.left)

self.insert\_to\_dll(node)

self.traverse\_tree(node.middle)

self.traverse\_tree(node.right)

def insert\_to\_dll(self, node):

if not hasattr(self, 'head'):

self.head = DLLNode(node.data)

self.tail = self.head

else:

new\_node = DLLNode(node.data)

self.tail.next = new\_node

new\_node.prev = self.tail

self.tail = new\_node

def print\_dll(self):

node = self.head

while node:

print(node.data)

node = node.next

# Example usage:

# Create a ternary tree

root = Node(1)

root.left = Node(2)

root.middle = Node(3)

root.right = Node(4)

root.left.left = Node(5)

root.left.middle = Node(6)

root.left.right = Node(7)

root.middle.left = Node(8)

root.middle.middle = Node(9)

root.middle.right = Node(10)

# Create a TernaryTree object and traverse the tree to create a doubly linked list

tree = TernaryTree(root)

tree.traverse\_tree(root)

# Print the doubly linked list

tree.print\_dll()

45. Convert arraylist into string

my\_list = ['hello', 'world', 'how', 'are', 'you']

my\_string = ' '.join(my\_list)

print(my\_string)

Output

hello world how are you

46. Python program to determine whether the given number is a Harshad Number

def is\_harshad(num):

"""

This function takes a number and returns True if it is a Harshad number, False otherwise.

"""

# calculate the sum of digits

digit\_sum = sum(int(digit) for digit in str(num))

# check if the number is divisible by the sum of its digits

if num % digit\_sum == 0:

return True

else:

return False

# example usage

num = 18

if is\_harshad(num):

print(num, "is a Harshad number")

else:

print(num, "is not a Harshad number")

47. Convert a set to stream

my\_set = {1, 2, 3, 4, 5}

stream = (x for x in my\_set)

for element in stream:

print(element)

Output

1

2

3

4

5

48. Python program to create a doubly linked list from a ternary tree

class Node:

def \_\_init\_\_(self, value):

self.value = value

self.prev = None

self.next = None

class TreeNode:

def \_\_init\_\_(self, value):

self.value = value

self.left = None

self.middle = None

self.right = None

def inorder\_traversal(node, prev):

if node is None:

return prev

prev = inorder\_traversal(node.left, prev)

new\_node = Node(node.value)

new\_node.prev = prev

if prev is not None:

prev.next = new\_node

prev = new\_node

prev = inorder\_traversal(node.middle, prev)

prev = inorder\_traversal(node.right, prev)

return prev

# example ternary tree

root = TreeNode(1)

root.left = TreeNode(2)

root.middle = TreeNode(3)

root.right = TreeNode(4)

root.middle.left = TreeNode(5)

root.middle.middle = TreeNode(6)

# create doubly linked list from ternary tree

head = inorder\_traversal(root, None)

# print doubly linked list

node = head

while node is not None:

print(node.value)

node = node.next

49. Write a program in python to check whether number is palindrom or not using recursive method?

def is\_palindrome\_recursive(n, temp):

if n == 0:

return temp

else:

temp = (temp \* 10) + (n % 10)

return is\_palindrome\_recursive(n // 10, temp)

# main function

num = int(input("Enter a number: "))

temp = is\_palindrome\_recursive(num, 0)

if temp == num:

print("The number is a palindrome")

else:

print("The number is not a palindrome")

50. Swap two numbers without using third variable

a = 10

b = 20

a = a + b

b = a - b

a = a - b

print("a after swap:", a)

print("b after swap:", b)

Output

a after swap: 20

b after swap: 10

51. Write a program to print all the prime numbers between two numbers

# function to check if a number is prime

def is\_prime(n):

# 1 is not a prime number

if n == 1:

return False

# 2 is a prime number

elif n == 2:

return True

# check if the number is divisible by any number other than 1 and itself

else:

for i in range(2, n):

if n % i == 0:

return False

return True

# input the range of numbers

start = int(input("Enter the start number: "))

end = int(input("Enter the end number: "))

# iterate over each number and print if it is prime

print("Prime numbers between", start, "and", end, "are:")

for i in range(start, end+1):

if is\_prime(i):

print(i)

52.Between 1 and 10 between 20 to 30

for i in range(1, 11):

print(i)

for j in range(20, 31):

print(j)

Output

1

2

3

4

5

6

7

8

9

10

20

21

22

23

24

25

26

27

28

29

30

53. Write a program to check the string is palindrome or not  
  
Madam wow cycle

def is\_palindrome(s):

# remove spaces and convert to lowercase

s = s.replace(" ", "").lower()

# compare the reversed string to the original string

return s == s[::-1]

# test the function

print(is\_palindrome("Madam")) # True

print(is\_palindrome("wow")) # True

print(is\_palindrome("cycle")) # False

54. 5. Write a program to print pattern?  
Input= 4 input =5  
  
\*\*\*\* \*\*\*\*\*  
\*\*\* \*\*\*\*  
\*\* \*\*\*  
\* \*\*  
\*

For input 4 the pattern would be:

\*\*\*\* \*\*\*\*

\*\*\* \*\*\*

\*\* \*\*

\* \*

For input 5 the pattern would be:

\*\*\*\*\* \*\*\*\*\*

\*\*\*\* \*\*\*\*

\*\*\* \*\*\*

\*\* \*\*

\* \*

n = int(input("Enter a number: "))

for i in range(n, 0, -1):

print("\*" \* i + " " \* (2\*n - 2\*i) + "\*" \* i)

55. Write a program to check the vowels in the string.  
Codoid innovations

string = "Codoid innovations"

# define a set of vowels

vowels = set("aeiouAEIOU")

# initialize a variable to count vowels

count = 0

# iterate through each character in the string

for char in string:

if char in vowels:

count += 1

# print the total count of vowels in the string

print("Total number of vowels in the string: ", count)

Output

Total number of vowels in the string: 8

56. Remove the duplicate elements in the array without using builtin function  
[ 5,4,10,20,4,6,10,39,4,39]

arr = [5, 4, 10, 20, 4, 6, 10, 39, 4, 39]

new\_arr = []

for num in arr:

if num not in new\_arr:

new\_arr.append(num)

print(new\_arr)

Output

[5, 4, 10, 20, 6, 39]

57. Find the largest number in the array (without using pre define functions)

# take an array as input from the user

arr = list(map(int, input("Enter an array of numbers separated by spaces: ").split()))

# initialize a variable to store the maximum value

max\_num = arr[0]

# iterate over the array to find the maximum value

for i in range(1, len(arr)):

if arr[i] > max\_num:

max\_num = arr[i]

# print the maximum value

print("The largest number in the array is:", max\_num)

58. Change the vowel characters to ”S”  
Python Pythsn

def replace\_vowels\_with\_s(string):

vowels = "aeiouAEIOU"

for char in vowels:

string = string.replace(char, "S")

return string

input\_string = "Python"

output\_string = replace\_vowels\_with\_s(input\_string)

print(output\_string)

Output

PythSn