**PRACTICAL-7**

**AIM:**

**a.** Write a program to demonstrate recursion in Python.

**Source Code:**

def factorial(n):

if n == 0:

return 1

else:

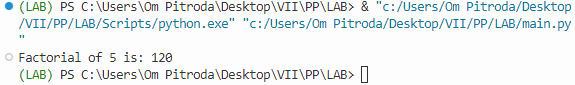
return n \* factorial(n - 1)

# Example usage:

result = factorial(5)

print("Factorial of 5 is:", result)

**Output:**



**b.** Create a function for Stack data structure in Python and implement necessary operations.

**Source Code:**

class Stack:

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

self.items = []

def push(self, item):

self.items.append(item)

def pop(self):

if not self.is\_empty():

return self.items.pop()

def is\_empty(self):

return len(self.items) == 0

def peek(self):

if not self.is\_empty():

return self.items[-1]

def size(self):

return len(self.items)

my\_stack = Stack()

my\_stack.push(1)

my\_stack.push(2)

my\_stack.push(3)

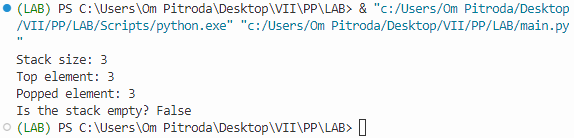
print("Stack size:", my\_stack.size())

print("Top element:", my\_stack.peek())

print("Popped element:", my\_stack.pop())

print("Is the stack empty?", my\_stack.is\_empty())

**Output:**



**c.** Create a function for Queue data structure in Python and implement necessary operations

**Source Code:**

from collections import deque

class Queue:

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

self.items = deque()

def enqueue(self, item):

self.items.append(item)

def dequeue(self):

if not self.is\_empty():

return self.items.popleft()

def is\_empty(self):

return len(self.items) == 0

def size(self):

return len(self.items)

# Example usage:

my\_queue = Queue()

my\_queue.enqueue(1)

my\_queue.enqueue(2)

my\_queue.enqueue(3)

print("Queue size:", my\_queue.size())

print("Dequeued element:", my\_queue.dequeue())

print("Is the queue empty?", my\_queue.is\_empty())

**Output:**

