2)

class BookStack:

def \_\_init\_\_(self, max\_size=100):

self.stack = []

self.max\_size = max\_size

def push(self, book\_title):

if len(self.stack) >= self.max\_size:

print("Stack Overflow! Cannot add more books.")

else:

self.stack.append(book\_title)

print(f'"{book\_title}" added to the stack.')

def pop(self):

if len(self.stack) == 0:

print("Stack Underflow! No books to retrieve.")

return None

else:

book\_title = self.stack.pop()

print(f'"{book\_title}" retrieved from the stack.')

return book\_title

def display(self):

if len(self.stack) == 0:

print("Stack is empty.")

else:

print("Books in stack (top to bottom):")

for book in reversed(self.stack):

print(f"- {book}")

library\_stack = BookStack()

library\_stack.push("To Kill a Mockingbird")

library\_stack.push("1984")

library\_stack.push("The Great Gatsby")

library\_stack.display()

retrieved = library\_stack.pop()

library\_stack.display()

4a)

SIZE = 5

queue = [None] \* SIZE

front = -1

rear = -1

def enqueue(value):

global rear, front

if rear == SIZE - 1:

print("Queue is FULL! Insertion is not possible!")

else:

if front == -1:

front = 0

rear += 1

queue[rear] = value

print(f"{value} enqueued to queue.")

def dequeue():

global rear, front

if front == -1 or front > rear:

print("Queue is EMPTY! Cannot dequeue.")

else:

removed = queue[front]

print(f"{removed} dequeued from queue.")

front += 1

if front > rear:

front = rear = -1

def display():

if front == -1 or front > rear:

print("Queue is EMPTY!")

else:

print("Queue elements are:")

for i in range(front, rear + 1):

print(queue[i])

while True:

print("\n--- Queue Operations Menu ---")

print("1. Enqueue")

print("2. Dequeue")

print("3. Display")

print("4. Exit")

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

if choice == '1':

value = input("Enter value to enqueue: ")

enqueue(value)

elif choice == '2':

dequeue()

elif choice == '3':

display()

elif choice == '4':

print("Exiting program. Goodbye!")

break

else:

print("Invalid choice. Please try again.")

4b)

class Node:

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

self.data = data

self.next = None

class Queue:

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

self.front = None

self.rear = None

def enqueue(self, value):

new\_node = Node(value)

if self.rear is None:

self.front = self.rear = new\_node

else:

self.rear.next = new\_node

self.rear = new\_node

print(f"{value} enqueued to queue.")

def dequeue(self):

if self.front is None:

print("Queue is EMPTY! Cannot dequeue.")

else:

removed = self.front.data

self.front = self.front.next

if self.front is None:

self.rear = None

print(f"{removed} dequeued from queue.")

def display(self):

if self.front is None:

print("Queue is EMPTY!")

else:

print("Queue elements are:")

temp = self.front

while temp is not None:

print(f"{temp.data} --> ", end="")

temp = temp.next

print("NULL")

queue = Queue()

while True:

print("\n--- Linked List Queue Menu ---")

print("1. Enqueue")

print("2. Dequeue")

print("3. Display")

print("4. Exit")

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

if choice == '1':

value = input("Enter value to enqueue: ")

queue.enqueue(value)

elif choice == '2':

queue.dequeue()

elif choice == '3':

queue.display()

elif choice == '4':

print("Exiting program. Goodbye!")

break

else:

print("Invalid choice. Please try again.")