**Exercise 5: Task Management System**

**Scenario:**

You are developing a task management system where tasks need to be added, deleted, and traversed efficiently.

**Steps:**

1. **Understand Linked Lists:**
   * Explain the different types of linked lists (Singly Linked List, Doubly Linked List).
2. **Setup:**
   * Create a class **Task** with attributes like **taskId**, **taskName**, and **status**.
3. **Implementation:**
   * Implement a singly linked list to manage tasks.
   * Implement methods to **add**, **search**, **traverse**, and **delete** tasks in the linked list.
4. **Analysis:**
   * Analyze the time complexity of each operation.
   * Discuss the advantages of linked lists over arrays for dynamic data.

**CODE:**

using System;

class Task

{

public int TaskId { get; set; }

public string TaskName { get; set; }

public string Status { get; set; }

public Task Next { get; set; }

public Task(int id, string name, string status)

{

TaskId = id;

TaskName = name;

Status = status;

Next = null;

}

}

class TaskList

{

private Task head = null;

public void AddTask(Task newTask)

{

if (head == null) head = newTask;

else

{

Task temp = head;

while (temp.Next != null) temp = temp.Next;

temp.Next = newTask;

}

}

public Task SearchTask(int id)

{

Task current = head;

while (current != null)

{

if (current.TaskId == id) return current;

current = current.Next;

}

return null;

}

public void TraverseTasks()

{

Task current = head;

while (current != null)

{

Console.WriteLine($"{current.TaskId}: {current.TaskName} [{current.Status}]");

current = current.Next;

}

}

public void DeleteTask(int id)

{

if (head == null) return;

if (head.TaskId == id) { head = head.Next; return; }

Task current = head;

while (current.Next != null)

{

if (current.Next.TaskId == id)

{

current.Next = current.Next.Next;

return;

}

current = current.Next;

}

}

}

class Program5

{

static void Main()

{

var list = new TaskList();

list.AddTask(new Task(1, "Design UI", "Pending"));

list.AddTask(new Task(2, "Implement Backend", "In Progress"));

list.AddTask(new Task(3, "Testing", "Pending"));

Console.WriteLine("All Tasks:");

list.TraverseTasks();

Console.WriteLine("\nSearching Task ID 2:");

var task = list.SearchTask(2);

Console.WriteLine(task != null ? $"{task.TaskName} - {task.Status}" : "Task not found");

list.DeleteTask(1);

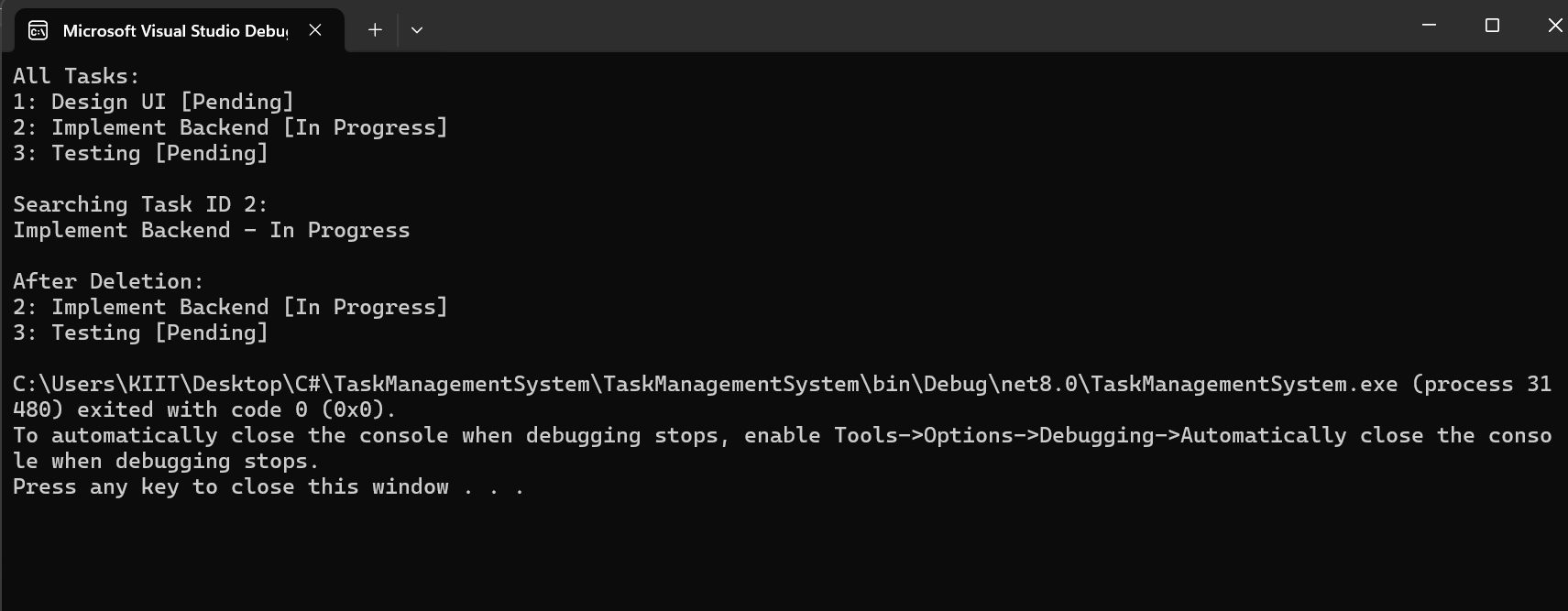
Console.WriteLine("\nAfter Deletion:");

list.TraverseTasks();

}

}

**OUTPUT:**



**ANALYSIS:**

**Time Complexities:**

* Add: O(n) (end), O(1) (start)
* Search: O(n)
* Traverse: O(n)
* Delete: O(n)

**Advantages over Arrays:**  
Dynamic size, efficient insert/delete. Better for frequent updates.