**Task Manager Assignment - Design Document (Due date: Feb 14, 2025 by 3:00 p.m.)**

## **Student Information**

* **Name: Gavin Juarez**
* **Student ID: 2022157044**
* **Course: Data Structures**
* **Date: 2/14/25**

## **Project Overview**

Provide a brief summary of your planned implementation. Describe the purpose of your task manager and how it will function.

**The Task Manager is a simple console application that helps users organize tasks. It will store tasks in a singly linked list and arrange them by priority (High → Medium → Low). Users can add, remove, search, and mark tasks as complete.**

## **Data Structures**

### **Task Class/Struct**

* Attributes: (List task description, priority, due date, completion status, and next pointer)
  + **Description (string): What the task is about.**
  + **Priority (string: High, Medium, Low): Determines task order.**
  + **Due Date (string or struct): Helps track deadlines.**
  + **Completion Status (bool): Shows if a task is done.**
  + **Next Pointer (Task):\* Links to the next task in the list.**
* Justification: (Explain why these attributes are needed)
  + **These attributes provide all necessary details for managing tasks efficiently, allowing operations like sorting, searching, and filtering.**

### **TaskList Class**

* **How will the linked list be managed?** (Head/Tail pointers, insertion logic)
  + **Head and Tail Pointers: The list will be managed using a head pointer for efficient insertion at the correct priority level. A tail pointer can help with faster appending if needed.**
  + **Insertion Logic: New tasks are placed based on priority.**
* **How will tasks be ordered?** (Priority-based, due date sorting, etc.)
  + **Priority-based Insertion: Higher-priority tasks come first. If two tasks have the same priority, they will be inserted in the order they were added.**

## **Core Functionalities**

### **Task Management**

* How will tasks be **added**, **removed**, and **marked as complete**?
  + **Adding Tasks: Inserted based on priority.**
  + **Removing Tasks: Completed tasks are deleted.**
  + **Marking Tasks as Complete: The completion status of a task will be updated when a user marks it complete.**
* How will tasks be **searched** by description?
  + **Searching Tasks: A task can be found by matching its description with user input.**

### **Display Operations**

* How will tasks be **displayed** and **filtered**?
  + **Displaying Tasks: Tasks will be shown in order of priority.**
  + **Filter by Priority: Displays tasks of a selected priority.**
* How will the system handle showing tasks due within **N days**?
  + **Showing Tasks Due in N Days: The program will compare due dates with the current date to display upcoming tasks.**
  + **Displaying Total Tasks: A counter will keep track of the total number of tasks.**

## **Error Handling & Edge Cases**

List potential errors (e.g., empty list operations, invalid inputs, memory allocation failures) and how they will be handled.

* **Empty List: Shows a message if there are no tasks.**
* **Invalid Input: Checks if the user enters wrong data.**
* **Memory Issues: Ensures proper memory allocation and deletion.**

## **Memory Management Plan**

Explain how you will prevent memory leaks and manage dynamic memory.

* **Avoiding Memory Leaks and manage dynamic memory.:**
  + **Dynamically allocated nodes will be properly deallocated upon deletion.**
  + **The destructor will ensure all nodes are freed when the program ends.**
  + **Efficient Insertion & Removal: Proper handling of pointers will maintain list integrity.**

## **User Interface Plan**

Describe how the menu system will work and how users will interact with the program.

* **Menu-Driven System: Users will interact with a numbered menu to choose operations like adding, deleting, searching, and displaying tasks.**
* **User Prompts: The program will guide users through inputs and confirm actions.**
* **Clear Formatting: Tasks will be displayed in a list format.**

## **Testing Strategy**

List planned test cases and expected outcomes.

| **Test** | **Outcome** |
| --- | --- |
| Add tasks with different priorities | Tasks are ordered correctly |
| Mark task as complete | Task is removed from the list |
| Search for a task | Task is found and shown |
| Filter tasks by priority | Only matching tasks are displayed |
| Show tasks due in N days | Tasks within range are listed |

## **Additional Features (if any)**

Mention any extra features you plan to implement beyond the base requirements.

**I don’t think I will be adding any additional features to the project.**

## 

## **Approval (Instructor Use Only)**

* **Approved:** Yes / No
* **Instructor Comments:**