****

Department of Computer Science,

CUI,Attock Campus.

**Program: BSE**

**Assignment # 01**

|  |  |
| --- | --- |
| **Student Name** | Laiba Zaib |
| **Registration #** | Sp23-bse-057 |
| **Section** | DS(thorey) |
| **Date** | 23 september,2024 |

**Professor: Sir Kamran**

### **Introduction:**

The C++ program presented is a **Task Management System** designed to handle tasks dynamically using a **singly linked list**. The tasks are represented by nodes in the linked list, and each task contains a unique task ID, a description, and a priority level. The core feature of the system is to manage the tasks based on their priority, ensuring that higher-priority tasks are handled first. The program also offers the functionality to add new tasks, view all tasks, remove the highest-priority task, and delete a task using its unique ID.

### **Purpose:**

The purpose of this program is to create a simple, efficient, and dynamic way of managing tasks based on their priority using a linked list. In many applications, task scheduling is crucial to ensure that more important tasks (higher priority) are handled first, which is essential in systems like operating systems, to-do lists, or even in project management tools. This program simulates how tasks are organized, inserted, and removed based on their priority, enabling users to focus on the most important tasks first.

### **Logic:**

**Data Structure (Node and Linked List):**

* 1. Each task is represented by a **Node** structure, containing the task ID (unique identifier), a task description (a brief string detailing the task), and the priority (an integer where higher numbers represent more important tasks).
  2. The linked list starts with a **head** pointer that points to the first task (Node) in the list. If there are no tasks, the head points to NULL.

**Task Insertion:**

* 1. When a new task is added, the program inserts it at the correct position based on its **priority**. The algorithm checks if the list is empty or if the new task has a higher priority than the first task, and if so, it inserts the task at the start.
  2. Otherwise, the program traverses the list until it finds the correct position (where tasks with higher priority come before those with lower priority). This ensures that when tasks are viewed or deleted, the highest-priority task is always the first one in the list.

**Task Deletion:**

* 1. The program allows the deletion of the **highest-priority task** by simply removing the first node (task) in the list.
  2. It also supports deleting a task by its unique **task ID**. The program traverses the list to find the task with the matching ID and removes it while maintaining the structure of the list.

**Task Viewing:**

* 1. The tasks are printed out in the order of their priority, starting from the head of the list (highest priority) to the end (lowest priority).

**Console-Based Menu:**

* 1. The program provides a console-based **menu** that allows users to interact with the system. Users can add new tasks, view all tasks, remove the highest-priority task, or remove a specific task by entering its ID. The menu operates in a loop until the user selects the option to exit the program.

### Detailed Steps:

**Add a new task**: The user inputs a unique task ID, a description, and the priority level. The program inserts the task in the appropriate place based on its priority.

**View all tasks**: The program displays all the tasks in the system, starting from the task with the highest priority.

**Remove the highest-priority task**: The task at the start of the list (which has the highest priority) is deleted.

**Remove a task by ID**: The user provides a task ID, and the program searches for and deletes the task with that ID.

**Exit**: The program terminates the loop and exits.

### Conclusion:

This **Task Management System** provides a simple and dynamic way to handle tasks efficiently based on their priority. The program leverages a **singly linked list** to dynamically manage tasks and offers key functionalities such as inserting tasks in priority order, viewing tasks, and removing tasks (either by highest priority or by task ID). The system can be further expanded for more complex task management systems or real-world applications, such as process scheduling, task queues, or priority-based to-do lists. The program is flexible enough to handle a variable number of tasks efficiently and ensures that users focus on the most important tasks at any given time.

