Comsats University Islamabad,
Attock Campus

Department : CS

Name: M.Ashtar Naqvi

Reg No.: SP23-BSE-026

Course: DS

Date: sep 24,2024

Assignment No.: 01

Submitted To: Mr. Kamran

Objective of the Assignment:

The objective of this assignment is to create a **Task Management System** using a **singly linked list**, where each task is represented as a node. The tasks include a unique ID, a description, and a priority level. The system supports operations like adding tasks in priority order, viewing tasks, removing the task with the highest priority, and removing tasks by their ID. The focus is on practicing linked list operations such as insertion, deletion, and traversal in C++.

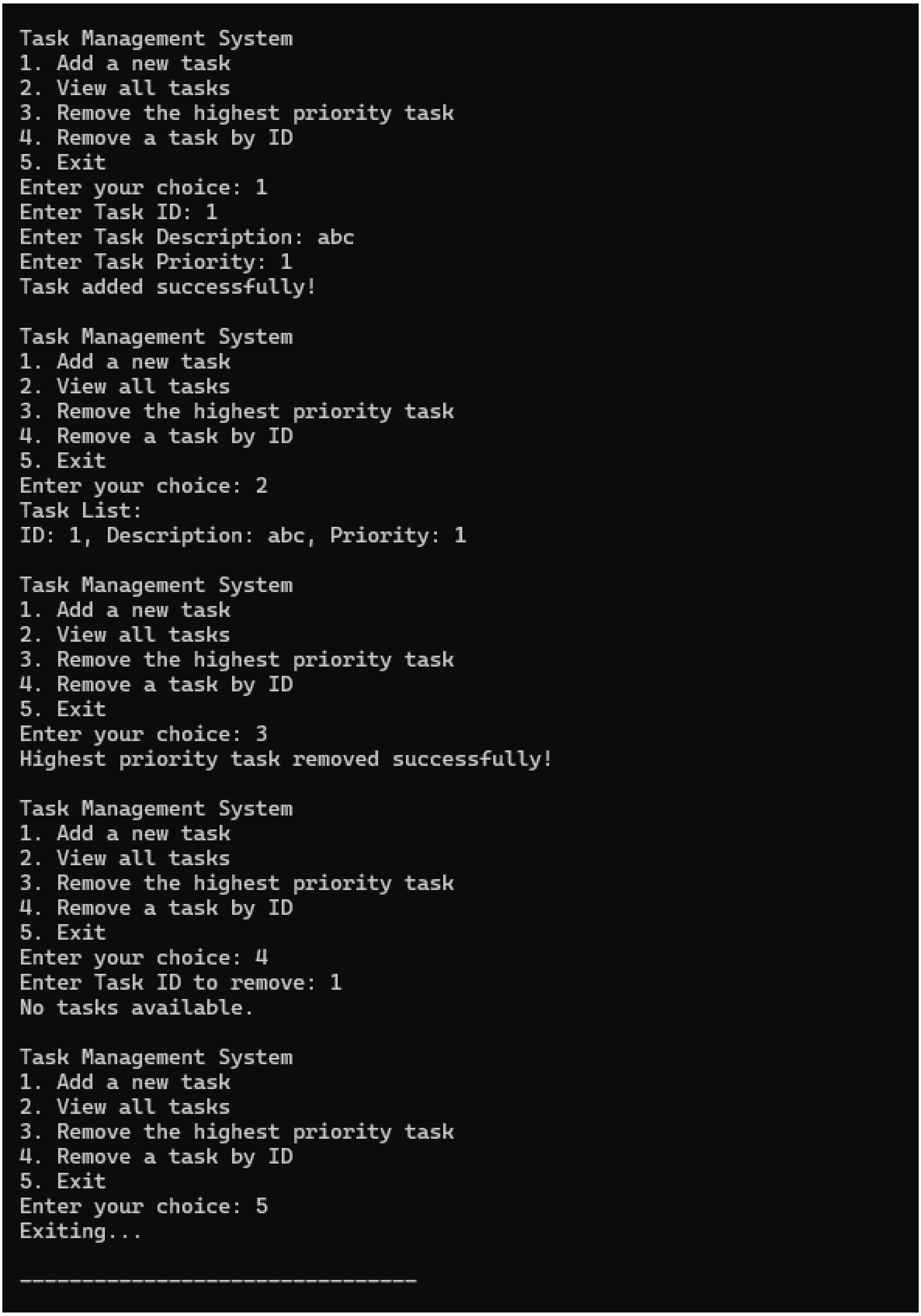
Operations Implemented:

1. **Add a Task**: Adds a new task to the list in such a way that tasks are ordered by their priority. Higher priority tasks are placed before lower priority tasks.
2. **View All Tasks**: Displays all tasks present in the linked list.
3. **Remove the Highest Priority Task**: Deletes the task with the highest priority, which is always the first node in the list.
4. **Remove a Task by ID**: Removes a specific task from the list using the task ID.

Code Explanation:

1. **TaskNode Structure**:
    - This defines the structure of a task, which contains a unique task ID (taskID), a description (description), a priority (priority), and a pointer to the next task node (next).
2. **createTask() Function**:
    - This function takes in a task ID, description, and priority, and returns a pointer to a new task node with the given data.
3. **addTask() Function**:
    - This function inserts a new task in the linked list while maintaining the order of tasks by their priority. If the list is empty or the task has the highest priority, the task is inserted at the head. Otherwise, it traverses the list to find the correct position for the new task.
4. **viewTasks() Function**:
    - This function traverses the linked list and prints out the task ID, description, and priority for each task in the list. If the list is empty, it displays a message indicating no tasks are available.
5. **removeHighestPriorityTask() Function**:
    - This function removes the task with the highest priority, which is always the first node in the list. It moves the head pointer to the next node and deletes the old head node.
6. **removeTaskByID() Function**:
    - This function removes a task by its task ID. It traverses the list looking for a task with the given ID. Once found, it adjusts the pointers and deletes the node from the list. If the task is not found, it prints an appropriate message.
7. **menu() Function**:
    - This function is responsible for displaying the console-based menu. The user can choose from options to add a task, view all tasks, remove the highest priority task, remove a task by ID, or exit the program.

Screenshot of code output:



```
Task Management System
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 1
Enter Task ID: 1
Enter Task Description: abc
Enter Task Priority: 1
Task added successfully!

Task Management System
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 2
Task List:
ID: 1, Description: abc, Priority: 1

Task Management System
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 3
Highest priority task removed successfully!

Task Management System
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 4
Enter Task ID to remove: 1
No tasks available.

Task Management System
1. Add a new task
2. View all tasks
3. Remove the highest priority task
4. Remove a task by ID
5. Exit
Enter your choice: 5
Exiting...

--------------------------------
```

Conclusion:

Through this assignment, I learned the practical implementation of singly linked lists in task management based on priorities. I exercised some vital linked list operations such as insertion, traversal and deleting nodes and how one can maintain order of the linked list. The hurdles that I encountered included how to make sure that the tasks were inserted in the right order, how to ensure the best of insertion is carried on without missing cases such as empty list to remove from or an id that is non-existent.

This project has helped me in a great way to possess the concepts internally of data structure, mainly the linked lists and realize the necessity of manipulating pointers in a dynamic data structure.