COMSATS UNIVERSITY ISLAMABAD (ATTOCK CAMPUSS)

PROGRAM BS-SE

NAME

AMIMA MASOOD

REG NO

SP23-BSE-038

COURSE

DS (THEORY)

ASSIGNMNET NO

#01

DATE

19 SEPTEMBER 2024

SUBMITTED TO

SIR MUHAMMAD KAMRAN

QUESTION NO # 01

Task Management System Implementation

CODE

```
#include <iostream>
#include <string>
Using namespace std;
struct Task {
   int id;
   string description;
   int priority;
   Task* next;
   Task(int taskId, const string& desc, int prio)
      : id(taskId), description(desc), priority(prio), next(nullptr) {}
};

class TaskManager {
private:
   Task* head;

public:
   TaskManager() : head(nullptr) {}

   // Add a new task in the correct position based on priority

   void addTask(int id, const string& description, int priority) {
      Task* newTask = new Task(id, description, priority);
      if (!head || head->priority < priority) {
         newTask->next = head;
         head = newTask;
      } else {
         Task* current = head;
         while (current->next && current->next->priority >= priority) {
            current = current->next;
         }
         newTask->next = current->next;
         current->next = newTask;
      }
      cout << "Task added: " << description << " with ID: " << id << "\n";
   }
```

```
// Remove the task with the highest priority

void removeHighestPriorityTask() {
   if (!head) {
      cout << "No tasks to remove.\n";
      return;
   }
   Task* temp = head;
   head = head->next;
   delete temp;
   cout << "Removed highest priority task.\n";
}
// Remove a specific task by ID

void removeTaskById(int id) {
   if (!head) {
      cout << "No tasks to remove.\n";
      return;
   }
   if (head->id == id) {
      Task* temp = head;
      head = head->next;
      delete temp;
      cout << "Removed task with ID: " << id << "\n";
      return;
   }
   Task* current = head;
   while (current->next && current->next->id != id) {
      current = current->next;
   }
   if (current->next) {
      Task* temp = current->next;
      current->next = current->next->next;
      delete temp;
      cout << "Removed task with ID: " << id << "\n";
   } else {
      cout << "Task with ID: " << id << " not found.\n";
   } }

// View all tasks

void viewTasks() const {
   if (!head) {
      cout << "No tasks available.\n";
```

```
            return;
        }
        Task* current = head;
        while (current) {
            cout << "ID: " << current->id << ", Description: " << current->description << ",Priority:" <<
current->priority  << "\n";
            current = current->next;
        }
    }
    ~TaskManager() {
        while (head) {
            removeHighestPriorityTask();
        }
    }
};
void displayMenu() {
    cout << "\n--- Task Manager Menu ---\n";
    cout << "1. Add a new task\n";
    cout << "2. View all tasks\n";
    cout << "3. Remove the highest priority task\n";
    cout << "4. Remove a task by ID\n";
    cout << "5. Exit\n";
}

int main() {
    TaskManager manager;
    while (true) {
        displayMenu();
        int choice;
        cin >> choice;

        switch (choice) {
            case 1: {
                int id, priority;
                string description;
                cout << "Enter task ID: ";
                cin >> id;
                cin.ignore(); // Clear newline from input buffer
                cout << "Enter task description: ";
                getline(cin, description);
                cout << "Enter task priority: ";
                cin >> priority;
                manager.addTask(id, description, priority);
                break;
```

```
        }
        case 2:
            manager.viewTasks();
            break;
        case 3:
            manager.removeHighestPriorityTask();
            break;
        case 4: {
            int id;
            cout << "Enter task ID to remove: ";
            cin >> id;
            manager.removeTaskById(id);
            break;
        }
        case 5:
            return 0;
        default:
            cout << "Invalid choice. Please try again.\n";
            break;
    }
  }
  return 0;

 // This line will never be reached due to the infinite loop above

}
```

OBJECTIVES OF THE ASSIGNMENT:

The objective of this program is to create a *Task Manager* that allows a user to manage tasks with various priorities. Tasks can be added , viewed, and removed, either based on their priority or specific task ID. The operations are performed via a simple menu-driven interface that continues to prompt the user until they decide to exit.

OPERATIONS IMPLEMENTED:

1. **Add a new task:**
   Adds a task to the list in a priority-based order.
2. **View all tasks:**
   Displays all tasks currently in the list.
3. **Remove the highest priority task:**
   Removes the task with the highest priority (the head of the list).

4. **Remove a task by ID:**

   Deletes a specific task by searching for its ID.

5. **Exit:**

   Terminates the program.

CODE EXPLANATION:

1. **Struct Task:**

   This is a simple structure representing a task, containing:

id: An integer to uniquely identify the task.

description: A string describing the task.

priority: An integer value representing the priority of the task (higher numbers indicate higher priority).

next: A pointer to the next task in the list, used to form a linked list.

The constructor initializes a task with an id, description, and priority, and sets the next pointer to nullptr.

2. **Class TaskManager:**

This class manages the tasks using a linked list. Each function in the class performs a specific task management operation.

3. **Constructor TaskManager():**

Initializes the head pointer to nullptr, meaning the task list starts out empty.

4. **Function addTask(int id, const std::string& description, int priority):**

This function creates a new task and inserts it into the list according to its priority. If the list is empty or if the new task has a higher priority than the current head, the task is inserted at the beginning.

Otherwise, the function traverses the list until it finds the appropriate position (where the next task has lower priority), and inserts the new task.

5. **Function removeHighestPriorityTask():**

This function removes the task with the highest priority, which is always at the head of the list. If the list is empty, it prints a message. If a task is found , it deletes the task and updates the head pointer to the next task.

6. **Function removeTaskById(int id):**

This function removes a task based on its unique id. It first checks if the task to be removed is the head. If not, it traverses the list until it finds the task with the matching id and removes it. If the task is not found, it prints a message.

7. **Function viewTasks():**

This function iterates through the task list and prints the id, description, and priority of each task. If the list is empty, it prints a message indicating that no tasks are available.

8. **Destructor ~TaskManager():**

This destructor ensures that all dynamically allocated tasks are deleted when the TaskManager object is destroyed. It repeatedly calls removeHighestPriorityTask() to clear the list.

9. Menu-driven interface (main() function):

The main function presents a menu to the user and handles the input for various operations. Depending on the user's choice, it invokes different functions from the TaskManager class to add, view, or remove tasks. The loop continues until the user chooses to exit by selecting option 5.

Conclusion

In conclusion, this C++ task management system showcases fundamental programming concepts such as data structures (linked lists), dynamic memory management, and user interaction through a console interface. The implementation is straightforward yet effective for managing tasks based on their priority levels. Users can easily add new tasks, view existing ones, and remove tasks as needed, all while maintaining an organized structure.This code serves as a great starting point for those looking to understand linked lists and task management systems in C++. Future enhancements could include features such as saving tasks to a file, loading tasks from a file upon startup, or even implementing additional sorting criteria beyond priority. Overall, this program demonstrates how to build a functional application in C++ while reinforcing key programming principles.

OUTPUT

main.cpp

Share | Run | Output | Clear

```cpp
#include <iostream>
#include <string>
class Node {
public:
    int taskID;
    std::string description;
    int priority;
    Node* next;
    Node(int id, const std::string& desc, int prio)
        : taskID(id), description(desc), priority(prio), next
            (nullptr) {}
};class TaskManager {
private:
    Node* head;
public:
    TaskManager() : head(nullptr) {}
    // Add a new task based on priority
    void addTask(int taskID, const std::string& description,
        int priority) {
        Node* newNode = new Node(taskID, description, priority
```

```
Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
Enter your choice (1-5): 1
Enter Task ID: 1
Enter Task Description: Finish report
Enter Task Priority (higher number means higher priority): 5

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
Enter your choice (1-5): 1
Enter Task ID: 2
Enter Task Description: Prepare presentation
```

main.cpp

Share | Run | Output

```cpp
        );
    if (head == nullptr || head->priority < priority) {
        newNode->next = head;
        head = newNode;
        return;
    }
    Node* current = head;
    while (current->next != nullptr && current->next
        ->priority >= priority) {
        current = current->next;
    }
    newNode->next = current->next;
    current->next = newNode;
}
// Remove the highest priority task
void removeHighestPriorityTask() {
    if (head == nullptr) {
        std::cout << "No tasks to remove." << std::endl;
        return;
    }
```

```
Enter Task Priority (higher number means higher priority): 3

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

Current Tasks:
Task ID: 1, Description: Finish report, Priority: 5
Task ID: 2, Description: Prepare presentation, Priority: 3

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
```

main.cpp

Share Run

```
    Node* removedTask = head;
    head = head->next;
    std::cout << "Removed task with ID: " << removedTask
        ->taskID
              << " and Description: " << removedTask
                  ->description << std::endl;
    delete removedTask;
}
// Remove a specific task by ID
void removeTaskByID(int taskID) {
    if (head == nullptr) {
        std::cout << "No tasks to remove." << std::endl;
        return;
    }
    if (head->taskID == taskID) {
        Node* removedTask = head;
        head = head->next;
        std::cout << "Removed task with ID: " <<
            removedTask->taskID
              << " and Description: " << removedTask
```


Output

```
Enter your choice (1-5): 3
Removed task with ID: 1 and Description: Finish report

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

Current Tasks:
Task ID: 2, Description: Prepare presentation, Priority: 3

Task Management System
1. Add a new task
2. View all tasks
3. Remove highest priority task
4. Remove a task by ID
5. Exit
```

main.cpp

Share Run

```
    return;
}
Node* current = head;
while (current->next != nullptr && current->next
    ->taskID != taskID) {
    current = current->next;
}
if (current->next == nullptr) {
    std::cout << "Task with ID " << taskID << " not
        found." << std::endl;
    return;
}
Node* removedTask = current->next;
current->next = current->next->next;
std::cout << "Removed task with ID: " << removedTask
    ->taskID
          << " and Description: " << removedTask
              ->description << std::endl;

delete removedTask;
```

Output

```
Removed task with ID: 2 and Description: Prepare presentation

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

Current Tasks:
No tasks available.

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