CODE:

```
#include <iostream>
#include<string>
using namespace std;

class Task {
private:
    string name;
    int priority;
    int execTime;
    Task* next;

    Task* head;

public:

    Task(string n = "", int p = 0, int e = 0) {
        name = n;
        priority = p;
        execTime = e;
        next = nullptr;
        head = nullptr;
    }

    // Insert based on priority (descending)
    void insertTask(string n, int p, int e) {
        Task* newTask = new Task(n, p, e);

        if (head == nullptr || p > head->priority) {
            newTask->next = head;
            head = newTask;
        } else {
            Task* current = head;
            while (current->next != nullptr && current->next->priority >= p) {
                current = current->next;
            }
            newTask->next = current->next;
            current->next = newTask;
        }
    }

    void displayTasks() {
        cout << "\nScheduled Tasks (Highest Priority First):\n";
        Task* current = head;
```

```
        while (current != nullptr) {
            cout << "Task: " << current->name
                 << ", Priority: " << current->priority
                 << ", Execution Time: " << current->execTime << " ms\n";
            current = current->next;
        }
    }

    // Execute tasks in order of priority (original list)
    void executeByPriority() {
        cout << "\nExecuting Tasks according to priority (higher priority first):\n";
        Task* current = head;
        while (current != nullptr) {
            cout << "Executing Task '" << current->name
                 << "' : " << current->execTime << " ms...\n";
            current = current->next;
        }
    }

    // Sort and execute tasks based on execution time (ascending)
    void executeByExecutionTime() {
        if (head == nullptr) return;

        Task* execHead = nullptr;
        Task* current = head;

        while (current != nullptr) {
            Task* nextTask = current->next;
            insertByExecutionTime(execHead, current);
            current = nextTask;
        }

        cout << "\nExecuting Tasks according to execution time(lower execution time will have
higher priority):\n";
        current = execHead;
        while (current != nullptr) {
            cout << "Executing Task '" << current->name
                 << "' : " << current->execTime << " ms...\n";
            Task* temp = current;
            current = current->next;
            delete temp;  // Free memory
        }

        cout << "\nAll tasks executed.\n";
```

```
    }

    // Insert into execution-time sorted list
    void insertByExecutionTime(Task*& execHead, Task* task) {
        task->next = nullptr;
        if (execHead == nullptr || task->execTime < execHead->execTime) {
            task->next = execHead;
            execHead = task;
        } else {
            Task* current = execHead;
            while (current->next != nullptr && current->next->execTime <= task->execTime) {
                current = current->next;
            }
            task->next = current->next;
            current->next = task;
        }
    }

    ~Task() {
        while (head != nullptr) {
            Task* temp = head;
            head = head->next;
            delete temp;
        }
    }
};

int main() {
    Task scheduler;
    int n;

    cout << "Enter number of tasks to schedule: ";
    cin >> n;

    for (int i = 0; i < n; ++i) {
        string name;
        int priority, execTime;

        cout << "\nTask " << i + 1 << " Name: ";
        cin >> name;

        cout << "Priority (higher = more important): ";
        cin >> priority;
```

```
        cout << "Execution Time (ms): ";
        cin >> execTime;

        scheduler.insertTask(name, priority, execTime);
    }

    scheduler.displayTasks();
    scheduler.executeByPriority();
    scheduler.executeByExecutionTime();

    return 0;
}
```

OUTPUT:

```
Enter number of tasks to schedule: 3

Task 1 Name: abc
Priority (higher = more important): 100
Execution Time (ms): 234

Task 2 Name: acn
Priority (higher = more important): 700
Execution Time (ms): 231

Task 3 Name: acnjd
Priority (higher = more important): 087
Execution Time (ms): 765

Scheduled Tasks (Highest Priority First):
Task: acn, Priority: 700, Execution Time: 231 ms
Task: abc, Priority: 100, Execution Time: 234 ms
Task: acnjd, Priority: 87, Execution Time: 765 ms

Executing Tasks according to priority (higher priority first):
Executing Task 'acn' : 231 ms...
Executing Task 'abc' : 234 ms...
Executing Task 'acnjd' : 765 ms...

Executing Tasks according to execution time(lower execution time will have higher priority):
Executing Task 'acn' : 231 ms...
Executing Task 'abc' : 234 ms...
Executing Task 'acnjd' : 765 ms...
```