Tutorial 4

Name: Shubham Santosh Dorake

PRN: B24CE1042

DIV: CE1-B

CODE:

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


class Task {
public:
   string taskname;
   int priority, exec_time;
   Task*next;


      Task(stringname,intp,inte)
         { taskname = name;
         priority = p;
      exec_time = e;
      next = NULL;
   }
};

Task* head = NULL;

//Insertnewtaskattheend(we'll sort later)
voidinsertTask(stringname,int p, int e) {
   Task*newnode=newTask(name, p, e);
```

```
    if (head == NULL) {
        head = newnode;


    } else {

        Task* temp = head;
        while(temp->next!=NULL)
            temp = temp->next;
        temp->next = newnode;

    }

}
//Bubblesortbypriority(descending)
voidsortByPriority(Task*head) {
    if (head == NULL) return;
    bool swapped;
    Task* ptr1;

    Task* lptr = NULL;

    do {

        swapped = false; ptr1
        = head;

        while (ptr1->next != lptr) {

            if (ptr1->priority < ptr1->next->priority) {
                swap(ptr1->taskname,
                ptr1->next->taskname); swap(ptr1->priority,
                ptr1->next->priority);
            swap(ptr1->exec_time,
            ptr1->next->exec_time); swapped = true;
        }
        ptr1 = ptr1->next;
        }
        lptr = ptr1;
```

```
    } while (swapped);
}
// Bubble sort by execution time (ascending)
void sortByExecutionTime(Task* head) {
    if (head == NULL) return;
    bool swapped;
    Task* ptr1;

    Task* lptr = NULL;

    do {

        swapped = false; ptr1
        = head;
        while (ptr1->next != lptr) {
              if (ptr1->exec_time > ptr1->next->exec_time)
                 { swap(ptr1->taskname,
                 ptr1->next->taskname); swap(ptr1->priority,
                 ptr1->next->priority);
                 swap(ptr1->exec_time,
                 ptr1->next->exec_time); swapped = true;
           }
           ptr1 = ptr1->next;
        }
        lptr = ptr1;
    } while (swapped);

}


    int main() {
        int n;
    cout<<"Enternumberoftaskstoschedule: ";
    cin >> n;


        for (int i = 0; i < n; i++) {
           string name;
```

```
    int p, e;

    cout << "\nTask " << i + 1 << " Name: ";
    cin >> name;
    cout << "Priority : ";
    cin >> p;
    cout << "Execution Time (in min) : ";
    cin >> e;

    insertTask(name, p, e);

}


sortByPriority(head);
cout<<"\nScheduledTasks(HighestPriority
First):\n"; Task* temp = head;

while (temp != NULL) {

    cout << "Task: " << temp->taskname
         << ", Priority: " << temp->priority
         <<",ExecutionTime:"<<temp->exec_time << "
    min\n"; temp = temp->next;

}



sortByExecutionTime(head);
cout<<"\nExecutingTasksaccordingtoexecution time (lower time → higher
priority):\n\n"; temp = head;
        while (temp != NULL) {
              cout<<"ExecutingTask"'<<temp->task
              name

            <<"':"<<temp->exec_time<<"min\n";

    temp = temp->next;
}
```

```
    cout << "\nAll tasks executed.\n";
    return 0;
}
```

OUTPUT:

```
Enter number of tasks to schedule: 3

Task 1 Name: STUDY
Priority : 3
Execution Time (in min) : 300

Task 2 Name: DRIVE
Priority : 5
Execution Time (in min) : 200

Task 3 Name: SLEEP
Priority : 1
Execution Time (in min) : 100

Scheduled Tasks (Highest Priority First):
Task: DRIVE, Priority: 5, Execution Time: 200 min
Task: STUDY, Priority: 3, Execution Time: 300 min
Task: SLEEP, Priority: 1, Execution Time: 100 min

Executing Tasks according to execution time (lower time → higher priority):

Executing Task 'SLEEP' : 100 min
Executing Task 'DRIVE' : 200 min
Executing Task 'STUDY' : 300 min

All tasks executed.
```