**Data Structures and Algorithms**

**Semester Project Deliverable 2**

**2023706; 2023725**

Project Title: Task Scheduler

Input and Output Design:

|  |  |
| --- | --- |
| INPUT | OUTPUT |
| Add task | Confirmation message |
| Remove task | Confirmation message |
| Display tasks | List of schedules and timeline |
| Invalid input | Error message |

Data Structures used:

1. Task (id/ description/ deadline/ priority/ completion time)

Stores all information about a task and sorts by priority and deadline.

1. Node (task/ next)

For a circular linked list of tasks.

Classes used:

1. TaskList (head/ tail/ size)

Head points to first and tail points to last node. Size contains current number of tasks.

1. TaskScheduler (tasklist/ taskqueue/ next\_id)

Priority queue schedules, while id is incremented automatically.

Functions used:

1. void addTask(const Task& task)

Adds a new task to end of our list and maintains circular nature by pointing tail to head.

1. bool removeTask(int id)

Removes a task by id set by user. Returns true if found and removed, otherwise returns false.

1. Task\* getTask(int id)

Returns a pointer to task with matching id provided.

1. void displayAll()

In a table form, this displays all tasks, with id, deadline, priority and all other info taken from user.

1. void setdeadline(id/ time\_t newDeadline)

Changes id of an already registered task and reorganizes schedule.

1. Void setpriority(int id/ int newpriority)

Updates priority of existing, specified id and reorganizes schedule.

1. Void setcompletiontime(int id/ int newcompletiontime)

Finds specified task and updates time.