1. **Node Class:**
   * Represents a node with attributes: **data** for job number, **priority** for priority level, **next**, and **prev** for pointers to the next and previous nodes.
2. **createNode Function:**
   * Creates a new node dynamically and initializes it with job number and priority.
3. **deque Function:**
   * Removes a node from the front of the queue (dequeues).
4. **enqueue Function:**
   * Adds a node to the end of the queue (enqueues).
5. **printQ Function:**
   * Prints the content of the queue in either forward or reverse direction.
6. **display Function:**
   * Displays the job execution sequence by printing queues with high, medium, and low priorities.
7. **addJob Function:**
   * Adds a job to the corresponding priority queue based on its priority.
8. **main Function:**
   * Initializes queues and flags for each priority level.
   * Enqueues jobs based on their priority.
   * Displays the initial job execution sequence.
   * Adds a new job with priority 'M' and displays the updated sequence.
   * Performs dequeue operations on jobs from each priority level.
   * Displays the final job execution sequence.
9. **Output Explanation:**
   * The code demonstrates the enqueue, dequeue, and display operations on jobs with different priority levels ('H', 'M', 'L').
   * Jobs are added to the corresponding priority queues based on their priority, and the sequence is displayed.

Algorithm:

1. \*\*Include necessary libraries and define the Node class.\*\*

2. \*Implement functions for enqueueing, dequeuing, and displaying the queues.\*

3. \*\*Implement the addJob function to add a job to the respective queue based on priority.\*\*

4. \*\*In the main function, create queues and schedule jobs based on priority.\*\*