Week6

1. In round robin, one-one packet from each queue is processed, a queue is skipped only if it is empty. If the queue is empty then we find the next non-empty queue and process it's packet. If all the queues are empty then we wait until a packet comes in any of the queue.
   Created a map 'que' whose key is "que_id" and value is queue of packets. First pushed every packet in its respective queue in the map. Round Robin will start from where the first packet arrived, so I found the minimum arriving time from all the queue in "que" map. Starting from this queue we will continue until all packets are processed.
   Now in processing, three cases arise:
   a. The arrival time of front packet of curr_que is less than or equal to curr_time, that means it has arrived on or before curr_time, then I processed it, and increased the curr_que, as now next queue is to be processed.
   b. The arrival time of front packet of curr_que is more than the curr_time, that means the curr_que is empty, then just increase the curr_que, i.e. check if packet has arrived in some other queue till curr_time.
   c. The arriving time of front packet of all the queues is more than curr_time, that is no packet has arrived yet and all the queues are empty, then we have to wait till the next packet arrives, i.e. the packet with minimum time from all the queue. The check for this case is done through the "flag" variable.

2. WFQ calculates the virtual finishing time for each packet, then processes the packet according to it, minimum virtual finishing time first.
So I pushed all the packets into a priority queue based on virtual finishing time. The priority queue in c++ by default is maximum priority queue, so I multiplied virtual finish time by -1 so that now it behaves as minimum priority queue. que_prev_time array is storing the finishing time of the previous packet of each queue.

Virtual time= max(previous finish time, arriving time) + length/weight
Then processed each packet one by one from the priority queue, the finishing time of the packet is calculated and printed.

Finishing Time= max(previous packet's time, arriving time) +
l length/service rate