Assignment 2 - Queues using Linked List Implementation

1 Queues Implementation

A queue differs from a stack in that its insertion and removal routines follow the first-in, first out (FIFO) principle Elements may be inserted at any time, but only the element, which has been in the queue the longest, may be removed. Elements are inserted at the rear (enqueued) and removed from the front (dequeued). So there are 2 main operations in queue Enqueue and Dequeue.

It's required to implement Queue using linked list implementation with the following functions: -

- **Initialize** It initializes queue, and make the head pointer equals NULL.
  Prototype: **Queue* initialize();**
- **Dequeue** It removes the first inserted element in the Queue and return it, and deletes its node.
  Prototype: **Process dequeue (Queue *q);**
- **Enqueue** It allocates a new node and attach it to the queue, then it inserts element at the end of the queue.
  Prototype: **void enqueue(Queue *q, Process value)**
- **isEmpty** It returns 1 if queue is empty or 0 otherwise.
  Prototype: **int isEmpty(Queue *q);-**

2 Application

One of the main functions of the operation system is to schedule processes that the processor executes. One of the very popular algorithms for this scheduling is called Round Robin, where each process enters a queue, executes in a FIFO order, and after a specified time its execution stops and returns back at the end of the queue if it still need time to continue execution or aborts if it has finished its required time of execution.

You will be given a file with the name of the processes that needs execution, time slot that it will enter the queue, and the needed time for its execution.

3 Example:

Input:

```
watching time slots = 15
P1 0  3
P2 0  2
P3 3  1
P4 10 3
```

Output:

```
P1   (0-->1)
P2   (1-->2)
P1   (2-->3)
P2   (3-->4) P2 aborts
P3   (4-->5) P3 aborts
P1   (5-->6) P1 aborts
idle (6-->7)
idle (7-->8)
idle (8-->9)
idle (9-->10)
P4   (10-->11)
P4   (11-->12)
P4   (12-->13) P4 aborts
idle (13-->14)
idle (14-->15)
stop
```

4 Algorithm

The given input file will have the watching time slots in the first line, then in the successive lines you will have the name of the process, time it enters the queue, and the needed time for execution respectively.
Your output should be the scheduling of the processes, showing when does each process abort and when does the processor is idle (no processes in the queue).
Each process should enter the queue whenever its execution time comes.
Each process will be grant 1 time slot on execution, if its execution time finishes it will abort, else it will be sent again to the queue and its execution time will be subtracted by 1.

5 Notes

- Implement your algorithms using c programming language.
- **You must follow the provided template.**
- You should work in groups of **two**.
- Copied assignments will be severely penalized.