Queue ADT (Linked List Implementations)

Purpose:
This worksheet is concerned with dynamic implementations of queues. You are not expected to complete the entire worksheet in class. Work on as many problems as you can; the remaining problems you can use for practice and to test your understanding of the application of queues.

1) Implement the abstract data type Queue. The operations to be implemented include `CreateQueue()`, `MakeEmptyQueue()`, `QueueSize()`, `Enqueue()`, `Dequeue()` and `IsEmptyQueue()`. Write a corresponding main program that uses these functions.

   You may use the following declaration for a dynamic queue:

```
struct Node
{
      int val;
      struct Node *next;
};

struct QueueRecord
{
      struct Node *front; /* pointer to front of queue */
      struct Node *rear; /* pointer to rear of queue */
      int size; /* number of items in queue */
};
typedef struct QueueRecord *Queue;
```

2) Store the queue and its functions in a separate file, create a header file **`queue.h`** and use this library of function in your main program.

3) Write a program that uses queue.h and gets integer values from the user until (s)he presses EOF. These integers will be stored in a queue. After that, write a function that, using only the queue ADT, calculates the sum and averages of the numbers in the queue without changing the content of the queue. Test this function by calling it from main and displaying the results on the screen.

4) Using only the functions in the queue ADT (i.e., queue.h), write a program that checks if the contents of two queues are the same. If they include the same elements, this function will return true, otherwise it will return false. Make sure that you test this function in the main function with appropriate function calls.

Please note that from this worksheet, you need to submit the solution of the following question (5)

5) Write a program that uses queue.h and simulate a simple book shop system as follows;
   - Each user has a unique id and request to get a book. Then, the robot may bring only one order of book at any given moment to the user.
   - Number of orders will be taken from the user.
   - Then for each order, user id will be created automatically, the starting time will be assigned randomly between 0 and 120 seconds (here make sure that starting time assigned in increasing order) and duration of bringing the book will be assigned randomly between 10 and 120 seconds.

Consider following data as an example:

User id	Starting time	Time to bring the book
1	20	60
2	30	65
3	32	63

This means that the user id 1 wants to start his order at time 20 which will take 60 s long for the robot to bring the book. Each input line represents a single user and contains the user's id, followed by the starting time, followed by the duration of bringing the book. The program should simulate the system and print a message containing the user ID, and the time whenever an order begins and ends. At the end of the simulation it should print the average waiting time for a preparation. (The waiting time is the amount of time between the time that the transaction was requested and the time it was started).

Tips: Create and get necessary inputs and create a queue of such records. You will then need to process each item in the queue.

Based on the given data example above, sample run should be as follows;

```
------- Simple book shop system-----
How many orders?3
Order ID 1
StartTime : 20
EndTime: 80
Waiting time: 0
Order ID 2
StartTime : 80
EndTime: 145
Waiting time : 50
Order ID 3
StartTime : 145
EndTime: 208
Waiting time : 113
The average waiting time of the system 54.33
```