```
//main
//Name:Mehmet Fatih Çelik
//ID: 2385268

#include <stdio.h>
#include <stdlib.h>
#include "queue.h"

int main(){
	srand(time(NULL));
	int orders, i;
	Queue q;
	q = CreateQueue();

	printf("------- Simple book shop system-----\n");
	printf("How many orders? ");
	scanf("%d",&orders);

	for(i=0; i<orders; i++)// Enqueue all the orders randomly, we didnt initialize userID here
		Enqueue(q);
	sortStartingTime(q); // we sorted our list based on the startingTime



	int ID = 1;
	struct Node *traversal = q->front->next;
	while(traversal){ // we initialized the userIDs in this while loop
		traversal->userID = ID;
		ID++;

		traversal = traversal->next;
	}
```

```
float totalWaiting = 0;
int waitingTime = 0;
int endTime;
int startTime;
for(i=0; i<orders; i++){
	if (i != 0){
		if(q->front->next->startingTime > endTime)
			startTime = q->front->next->startingTime;
		else
			startTime = endTime;

		endTime = startTime + q->front->next->bringingTime;

		waitingTime = endTime - q->front->next->startingTime;
		if (waitingTime < 0)
			waitingTime = 0;
		totalWaiting += waitingTime;



		printf("Order ID %d\n",q->front->next->userID);
		printf("StartTime : %d\n",startTime);
		printf("EndTime: %d\n",endTime);
		printf("Waiting time: %d\n",waitingTime);
		Dequeue(q);
	}
	else{// if i == 0
	printf("Order ID %d\n",q->front->next->userID);
	printf("StartTime : %d\n",q->front->next->startingTime);

	endTime = q->front->next->bringingTime + q->front->next->startingTime;
```

```
            printf("EndTime: %d\n",endTime);
            printf("Waiting time: %d\n",waitingTime);
            Dequeue(q);
            }
        }

        printf("The average waiting time of the system %.2f",totalWaiting/orders);

        return 0;
}
```

```
//queue.c
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"

Queue CreateQueue(void){
	Queue q;
	q = (struct QueueRecord*)malloc(sizeof(struct QueueRecord));
	if (q == NULL){
		printf("Out of memory!");
		exit(-1);
	}

	q->size = 0;
	q->front = (struct Node*)malloc(sizeof(struct Node));
	if (q->front == NULL){
		printf("Out of memory!");
		exit(-1);
	}

	q->front->next = NULL;
	q->rear = q->front;
	return q;
}

void Enqueue(Queue q){
	struct Node *t;
	t = (struct Node*)malloc(sizeof(struct Node));
	t->next = NULL;
	t->startingTime = rand()%121;
	t->bringingTime = 10 + rand()%111;
```

```
        q->rear->next = t;
        q->rear = t;
        q->size++;
}

void sortStartingTime(Queue q){
        int swapped, temp;
        struct Node *t = NULL;
        struct Node *t2;

        do{
                swapped = 0;
    t2 = q->front->next;

    while (t2->next != t){
        if (t2->startingTime > t2->next->startingTime){
                temp = t2->startingTime; // swapping startingTime
                t2->startingTime = t2->next->startingTime;
                t2->next->startingTime = temp;

                temp = t2->bringingTime; // swapping bringingTime
                t2->bringingTime = t2->next->bringingTime;
                t2->next->bringingTime = temp;

                swapped = 1;
                        }
                        t2 = t2->next;
                }
                t = t2;

        }while(swapped);
```

```
}


void Dequeue(Queue q){
        struct Node *removal;
        removal = q->front->next;
        q->front->next = removal->next;
        free(removal);
        q->size--;
        if (q->size == 0)
                q->rear = q->front;
}
```

```
//queue.h
struct Node{
	int userID;
	int startingTime;
	int bringingTime;
	struct Node *next;
};

struct QueueRecord{
	struct Node *front;
	struct Node *rear;
	int size;
};
typedef struct QueueRecord *Queue;

Queue CreateQueue(void);
void Enqueue(Queue);
void sortStartingTime(Queue);
void Dequeue(Queue);
```