CSE 3010 – Data Structures & Algorithms

Lecture #22

What will be covered today

- Mid-semester examination pattern
- Implementation of queue data structure

Mid-semester examination pattern

Date of Examination	27th January 2020
Time	9:00 am
Duration	2 hours
Format	Open book
Type	Problem solving
Through	LMS
Syllabus	Until **Queue data structure**

Queue - Implementation using arrays

```
QUEUE* create(QUEUE *q) {

    q = (QUEUE *) malloc(sizeof(QUEUE));

    if (q != NULL) {
        q->front_queue = -1;
        q->rear_queue = -1;
        return q;
    }
    else
        return NULL;
}
```

Queue - Implementation using arrays

```
int add(QUEUE *q, ITEM item) {

   if (!isFull(q)) {
        if (q->front_queue == -1)
            q->front_queue = 0;
        q->rear_queue = q->rear_queue + 1;
        q->queue[q->rear_queue] = item;
        return 1;
    }
    else
        return 0; // When QUEUE is full
}
```

Queue - Implementation using arrays

```
ITEM delete(QUEUE *q) {

    ITEM removedItem;

    if (!isEmpty(q)) {
        removedItem = q->queue[q->front_queue];
        q->front_queue = q->front_queue + 1;
    }
    else
        removedItem.appln_name[0] = '\0';
    return removedItem;
}
```

Queue - Implementation using arrays

```
ITEM front(QUEUE *q) {

    if (!isEmpty(q))
        return q->queue[q->front_queue];
    else {
        ITEM temp;
        temp.appln_name[0] = '\0';
        return temp; //
    }
}
```

Queue - Implementation using arrays

```
bool isEmpty(QUEUE *q) {

    if ((q->rear_queue - q->front_queue) == -1)
        return true;
    else
        return false;
}

bool isFull(QUEUE *q) {

    if (q->rear_queue == SIZE-1)
        return true;
    else
        return false;
}
```