EECE.3220: Data Structures

Key Questions

Queues (Lectures 22-25)

QUESTIONS

1. Describe the general design of a queue data structure, as well as some basic applications in which it is useful.
2. Describe how an array can be used to implement a queue class.
3. Explain how a linked queue is implemented.
4. Describe the algorithms for each of the following linked Queue operations:
   a. Construction
   b. Destruction
   c. Checking if queue is empty
   d. Reading front of queue
   e. Enqueueing
   f. Dequeueing

EXAMPLES

Write definitions for each function below, assuming the following Queue definition:

```
template <class T>
class Queue {
public:
   Queue(unsigned maxSize = 1024);     // Constructor
   ~Queue();                           // Destructor
   bool empty() const;                 // Returns true if queue empty
   void enqueue(const T &val);         // Add val to tail of queue
   void dequeue();                     // Remove head of queue
   T front();                          // Read data of head of queue
private:
   T *list;                 // The actual data stored in the queue
   int front, back;         // Indexes for head & tail of queue
   unsigned cap;            // Capacity (max size) of queue
};

// Default constructor
template <class T>
Queue::Queue(unsigned maxSize = 1024)
{

}

// Destructor
template <class T>
Queue::~Queue()
{


}

// True if list is empty
template <class T>
bool Queue::empty() {


}
```

```
// Add new value to back of queue
template <class T>
void Queue::enqueue(const T &val) {









}
// Remove element at front of Queue
template <class T>
void Queue::dequeue() {









}

// Retrieve value of element at top of Queue
template <class T>
T Queue::front() {









}
```