EECE.3220: Data Structures

Key Questions

Finishing queues; linked lists (Lecture 25)

QUESTIONS

1. Describe how standard containers can be used to implement a queue.
2. Describe the properties of a linked list.
3. Explain the algorithms for inserting and deleting data from an ordered linked list.
4. Describe doubly-linked lists.

EXAMPLES

What does the following program print?

```
int main() {
   Stack S;
   Queue Q;

   int list[10] = {3, 10, 0, -5, 9,
                   78, 6, 3220, 1, 1146};

   for (int i = 0; i < 10; i = i + 2)
      S.push(list[i]);

   while (!S.empty()) {
      cout << "S " << S.top() << "\n";
      Q.enqueue(list[S.top()]);
      S.pop();
   }

   while (!Q.empty()) {
      cout << "Q " << Q.front() << "\n";
      Q.dequeue();
   }
   return 0;
}
```

Given the linked list definition below (which is not a class template, but could easily be changed to be):

```
class LList {
public:
   LList();                    // Default constructor
   ~LList();                   // Destructor
   bool empty();               // True if list is empty
   void insert(int v);         // Add new value to list
   void remove(int v);         // Remove node with v
private:
   class Node {
   public:
      int val;       // Value in each node—could be general
      Node *next;   // Pointer to next node
   };

   Node *first;     // Pointer to first node
};
```

First, explain the point of defining a Node class inside the LList class. Then, write the member function definitions:

```
// Default constructor
LList::LList() {



}

// Destructor
LList::~LList() {









}

// True if list is empty
bool LList::empty() {



}
```

```
// Add new value to list
void LList::insert(int v) {









}

// Remove node with v
void remove(int v) {









}
```