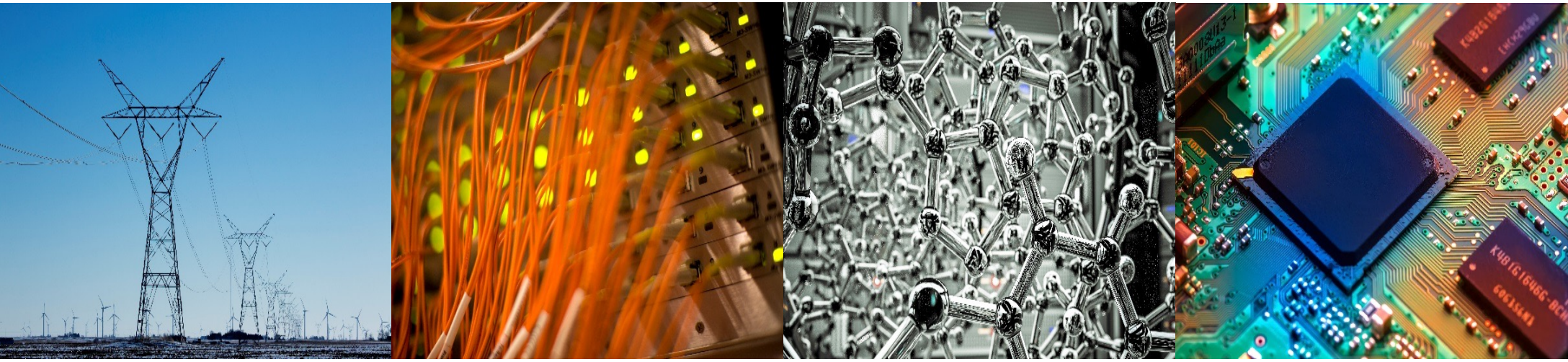
ECE 220 Computer Systems & Programming

Lecture 27 – Course Review Day 2

May 5, 2026

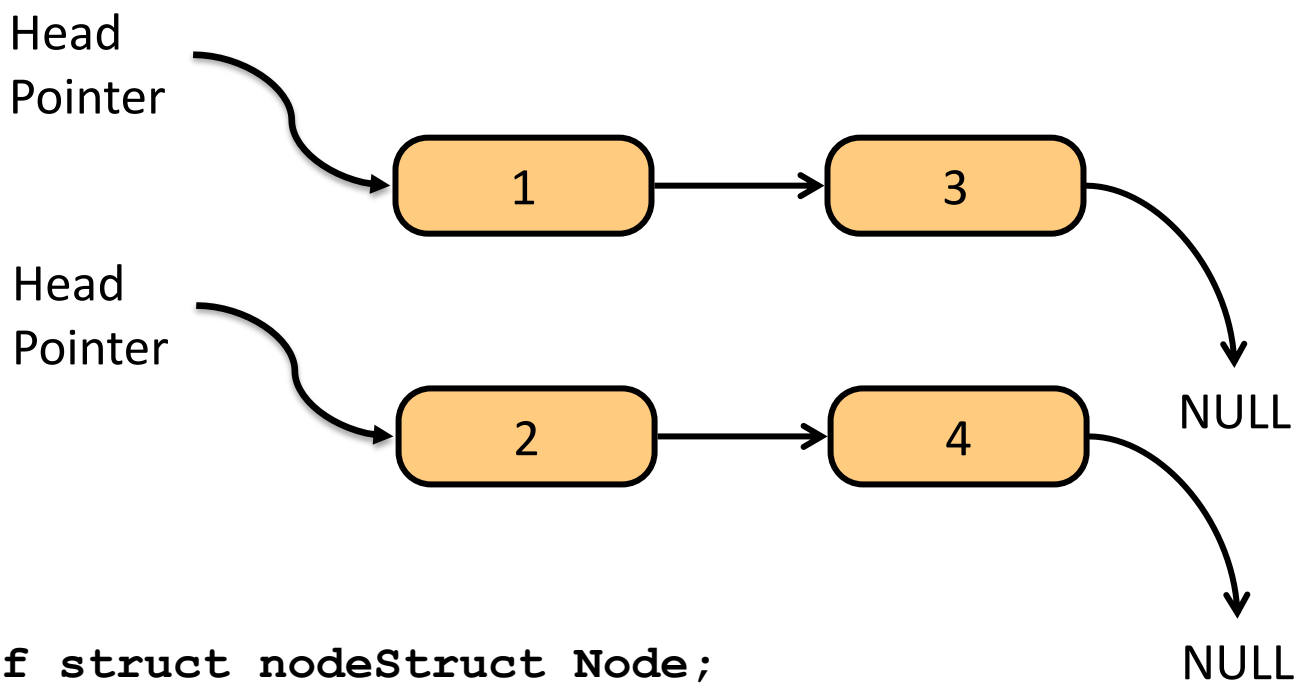


- **Programming competition: 5/6, 6pm – 8pm in ECEB 1015**
- **Final exam conflict request is due on 5/6**
- **HKN review session: Sunday, 5/10, 12:30pm – 3pm in ECEB 1002**

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Exercise: Linked List

- Given two sorted linked lists, merge them as one sorted linked list.



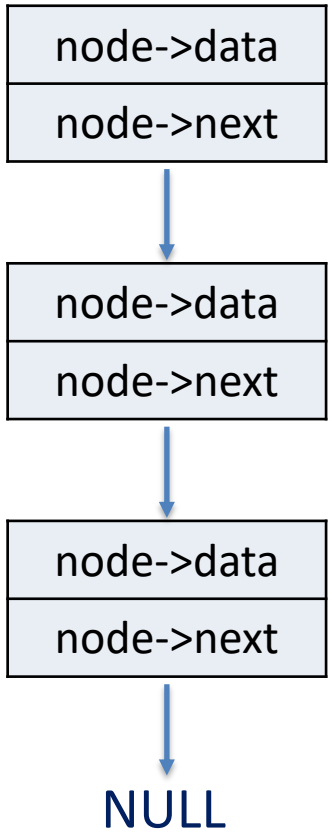

```
typedef struct nodeStruct Node;
struct nodeStruct{
    int data;
    Node *next;
};
```

Recursive Implementation

```
Node *mergeList(Node *L1, Node *L2){
   /* base cases: either L1 or L2 is NULL */




   /* recursive cases:
      (1) if data of head node in L1 is less than that in L2
      (2) otherwise */




}
```

C to LC-3 Conversion

L1 starts at x4000, L2 starts at x5000



R1: L1 **R3**: L1->data / L1->next

R2: L2 **R4**: L2->data / L2->next

```
MERGE_LIST
;;Part 1 - callee build up
;allocate space for bookkeeping info


;store return address to stack


;store caller's frame pointer


;set up new frame pointer


;;Part 2 - implement function logic
;if(L1 == NULL)



;if(L2 == NULL)
```

```
;if(L1->data < L2->data)
;get L1->data and L2->data




;calculate L1->data - L2->data




;caller build up for mergelist(L1->next, L2)
```

```
;caller tear down for mergelist(L1->next, L2)
```

```
ELSE
;else
;caller build up for mergelist(L1, L2->next)

;caller tear down for mergelist(L1, L2->next)
```

```
RETURN_L1



RETURN_L2



;;Part 3 – callee tear down
DONE
```

Exercise: Binary Tree

Given a binary tree and a target value, determine whether there's a path from root to a leaf node that will sum up to the target value.

```
typedef struct treeNode tnode;
struct treeNode{
        int data;
        tnode *left;
        tnode *right;
};
```

Algorithm:

1. If root is null, return false
2. If we reach a leaf node, check whether the current path sum matches the target and return accordingly
3. Recursively call the function on the left and right subtrees with the 'new' target (deduct current node's value from previous target). Return the result accordingly.

```
bool path_sum(tnode *root, int target){
      /* base case: if root is NULL, return false */



      /* check if current node is a leaf */



      /* recursively call the function on left and right subtrees */



}
```