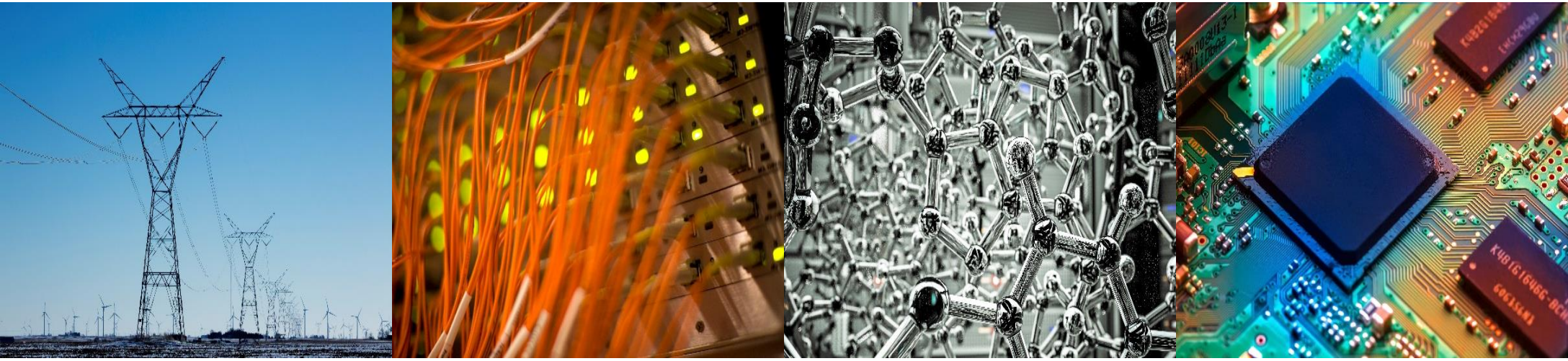
ECE 220 Computer Systems & Programming

Lecture 27 – Course Review Day 2

December 10, 2024



- **Programming competition is tomorrow, 12/11, at 7pm in ECEB 1013**
- **Final exam conflict request is due on 12/11**

ILLINOIS
Electrical & Computer Engineering
GRAINGER COLLEGE OF ENGINEERING

Exercise: Linked List

Given a singly linked lists of integers

1) remove all nodes that contain a certain value

2) split the list into two smaller lists

Example

Given:	10 → 1 → 4 → 7 → 1 → 2 → 9
After removal of nodes with the value '1':	10 → 4 → 7 → 2 → 9
After splitting into two smaller lists:	10 → 4 → 7 , 2 → 9

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

Remove Nodes in a Singly Linked List

```
void remove_nodes(Node **list, int value){
/* handle empty(or end of) list */


/* found node with matching data */
if(                                                    ){




}
/* otherwise */
else{

}
}
```

Split List Into Two Smaller Lists

```
void split_list(Node *head, Node **front, Node **back){
/* use slow and fast pointers to find the 'mid' point */
Node *slow, *fast;

/* if the list has less than 2 nodes */
if(                                                    ){




}
```

```
/* otherwise (at least 2 nodes) */
else{
   slow = head;
   fast = head->next;
   while(                                                    ){




   }
   *front =

   *back =

   slow->next =
}

}
```

Exercise: C to LC-3 Linked List Traversal

```
typedef struct nodeStruct node;
struct nodeStruct{
   char data;
   node *next;
};

void print_list(node *head){
   if(head != NULL){
      printf("%c ", head->data);
      print_list(head->next);
   }
}
```

```
;data.asm

.ORIG x4000

.FILL x55
.FILL x4002

.FILL x49
.FILL x4004

.FILL x55
.FILL x4006

.FILL x43
.FILL x0

.END
```

```
.ORIG x3000
MAIN  LD R5, RSTACK     ;set R5 to the bottom of the stack
      LD R6, RSTACK     ;set R6 to the same address
      LD R0, HEAD       ;load head pointer to R0
      STR R0, R6, #0    ;caller set-up: push head pointer to stack
      JSR PRINT_LIST
      ADD R6, R6, #2    ;caller tear-down
      HALT
HEAD .FILL x4000
RSTACK .FILL x7000

PRINT_LIST
;;Part 1 - callee set-up: push bookkeeping info
```

```
;;Part 2 - implement function logic
;if(head == NULL) skip to TEAR_DOWN;


;printf("%c ", head->data);


;print_list(head->next);
;;;caller set-up and tear-down of recursive call




;;Part 3 - callee-teardown: part of activation record
TEAR_DOWN


RET
.END
```

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 */



}
```