Programming Studio #13

ECE 190

Programming Studio #13

- Concepts this week:
  - Singly linked lists

What is a Singly Linked List

- A linked list is a series of "nodes".
- Each node has data, and a pointer to the next node.
- Lists can be of any size, and each "chain link" is allocated when needed.

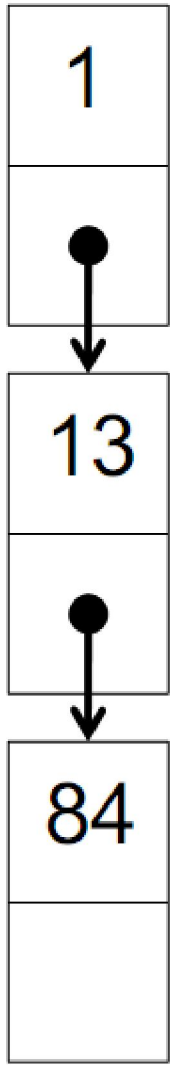
```
struct Node {
    int Value;
    struct Node *next_ptr;
};
```

Create a Linked List

- Let's make a linked list that has three values stored it in.
- Let each one point to the next one, and let the last node have a NULL pointer.
- As we create new lists, insert them into the list.

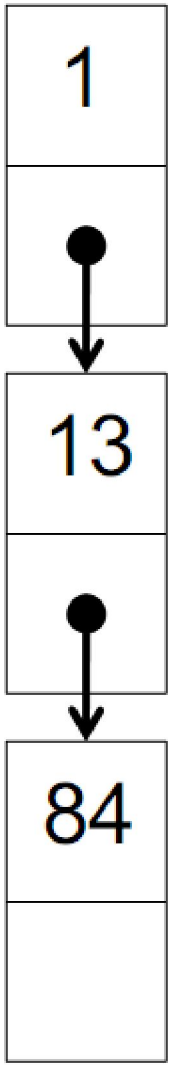
Example Code

```
struct Node *one_ptr = NULL;
struct Node *two_ptr = NULL;
struct Node *three_ptr = NULL;

one_ptr = malloc (sizeof (strcut Node));
one_ptr -> Value = 1;

two_ptr = malloc (sizeof (struct Node));
two_ptr -> Value = 13;
one_ptr -> next_ptr = two_ptr;

three_ptr = malloc (sizeof (struct node));
three_ptr -> Value = 84;
two_ptr -> next_ptr = three_ptr;
```



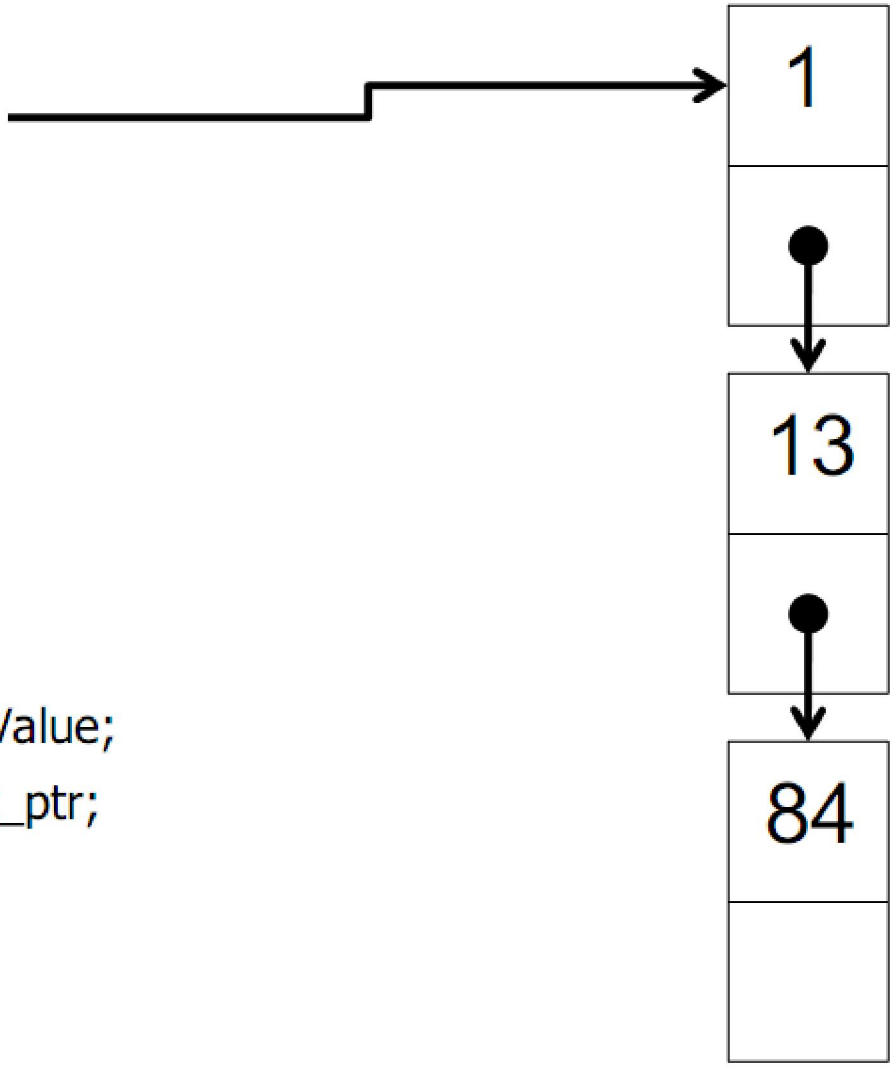
Traversing Linked List

- How do we travel through a list of any size?
- Let us start with one pointer, a "head" pointer.



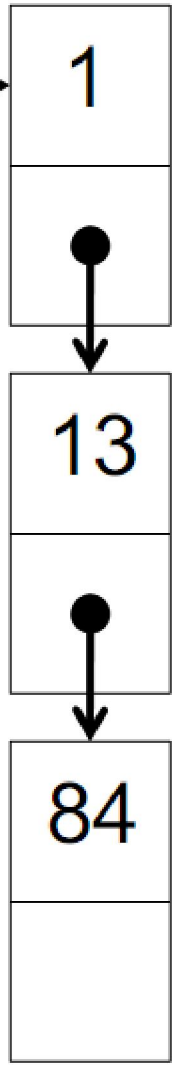
Traversing Linked List

```
struct Node *head_ptr = NULL;

/* Make our three item list */
makeList (&head_ptr);

/* Print all values in list */
while (head_ptr != NULL)
{
    printf ("%d\n", head_ptr -> Value;
    head_ptr = head_ptr -> next_ptr;
}

/* Does this really work? */
```



Traversing Linked List

```
/* No, because we cannot change the
     pointer back to the beginning! */
struct Node *head_ptr = NULL;
struct Node *list_ptr = NULL;

/* Make our three item list */
makeList (&head_ptr);
list_ptr = head_ptr;

/* Print all values in list */
while (list_ptr != NULL)
{
    printf ("%d\n", list_ptr -> Value;
    list_ptr = list_ptr -> next_ptr;
}
```



Inserting Items

- In previous examples, we just hard-coded the creation of a list.
- But we need to be able to handle any list size.
- Solution: Write code to insert items one at a time.

```
void insertList (
    struct Node *head_ptr,
    struct Node *new_ptr)
{
    /* Iterate through all items */
    <code>

    /* Change tail pointer from NULL
     * to new_ptr.
     */
    <code>
}
```

Inserting Items

- What happens when head_ptr is NULL?
- Need this code to handle this special case.

```
void insertList (
    struct Node *head_ptr,
    struct Node *new_ptr)
{
    /* Iterate through all items */
    <code>

    /* Change tail pointer from NULL
     * to new_ptr.
     */
    <code>
}
```

Inserting Items

- Solution: Pass in pointer pointing to head_ptr!
- Be careful, only modify pointer to head pointer if head pointer is NULL.
- Does your head hurt yet?
  - (mine does)

```
void insertList (
    struct Node **head_ptr,
    struct Node *new_ptr)
{
    struct Node *temp = *head_ptr;
    /* Special case */
    <code>

    /* Iterate through all items */
    <code>

    /* Change tail pointer from NULL
     * to new_ptr.
     */
    <code>
}
```

PSMP13

- Download psmp13.c from the website.
- Implement function code for inserting, deleting, and printing all items in a linked list.
- Modify the code to use doubly linked lists for fun.