BITS Pilani
Hyderabad Campus

CS F111: Computer Programming

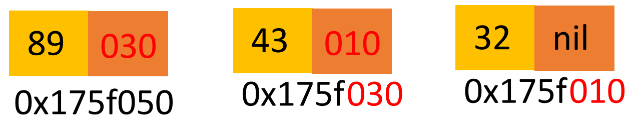
(Second Semester 2020-21)

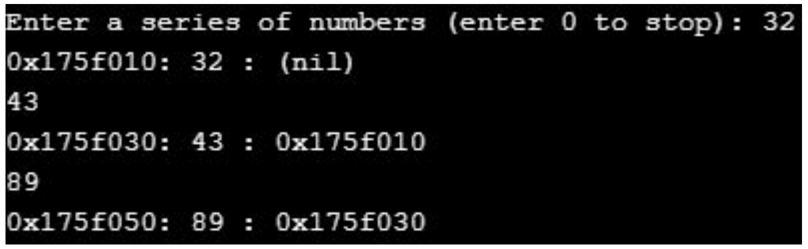
Lect 30: Linked List contd..

Nikumani Choudhury
Asst. Professor, Dept. of Computer Sc. & Information System

Inserting into Linked List: Ex

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   struct node {
       int data;
       struct node *next;
  };
   struct node *first = NULL, *temp;    int n;
   printf("Enter a series of numbers (enter 0 to stop): ");
   scanf("%d", &n);
   while (n != 0) {
       temp = malloc(sizeof(struct node));
       temp->data = n;
       temp->next = first;
       first = temp;
       printf ("%p: %d : %p\n", temp, temp->data, temp->next);
       scanf("%d", &n);}
return 0;}
```





```
Enter a series of numbers (enter 0 to stop): 32
0x175f010: 32 : (nil)
43
0x175f030: 43 : 0x175f010
89
0x175f050: 89 : 0x175f030
```

Insertion at Beginning/ Middle ?

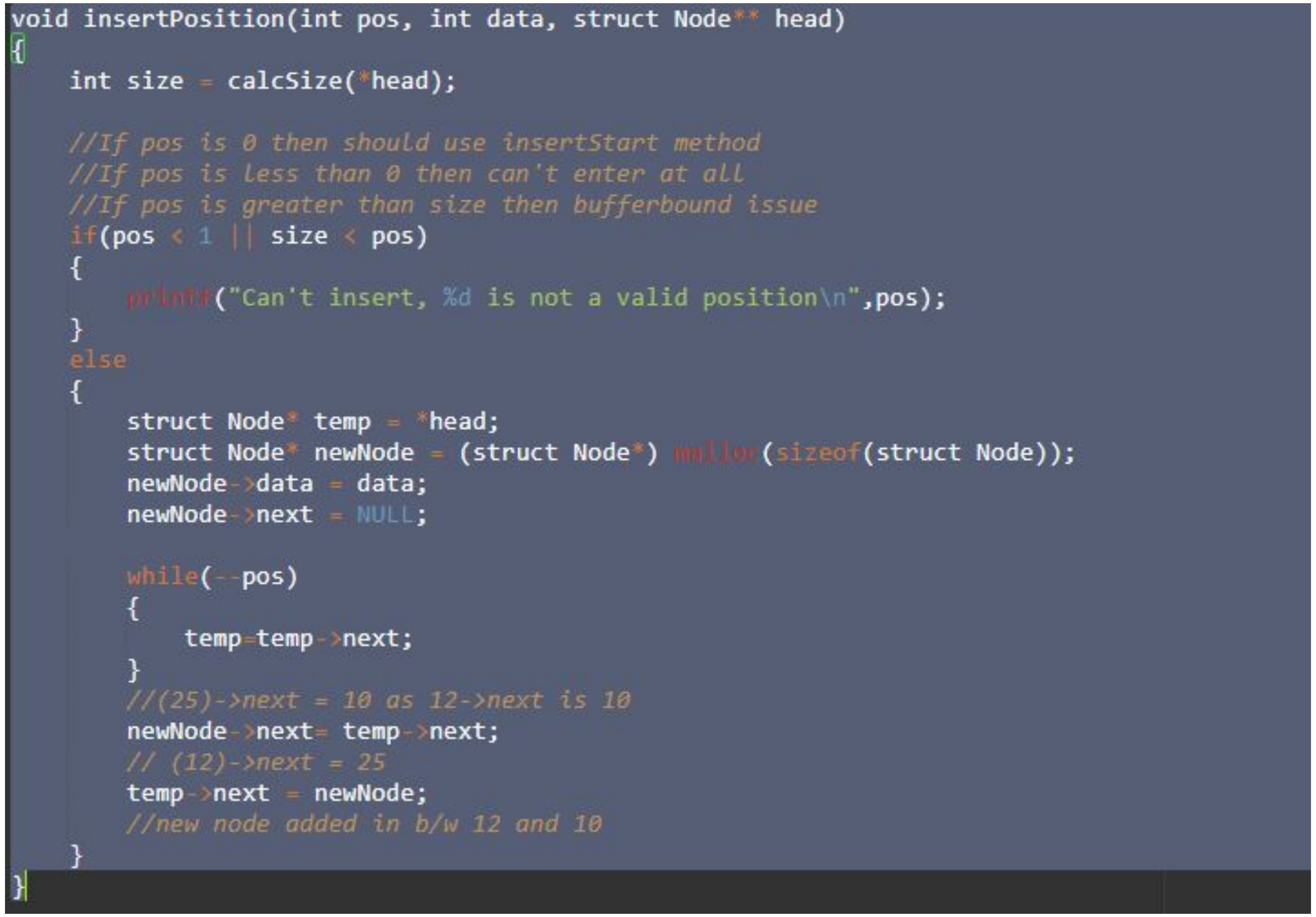
Insertion at beginning

```
void insertStart(struct Node** head, int data)
{    struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
     newNode->data = data;
     newNode->next = *head;
//changing the new head to this freshly entered node
   *head = newNode;
}
```

Insertion at the end

```
void insertLast(struct Node** head, int data)
{
        struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
        newNode->data = data;
        newNode->next = NULL;
        //need this if there is no node present in linked list at all
        if(*head==NULL)
        {       *head = newNode;
                return;
        }
        struct Node* temp = *head;
        while(temp->next!=NULL)
                temp = temp->next;
        temp->next = newNode;
}
```

Insertion at nth Position

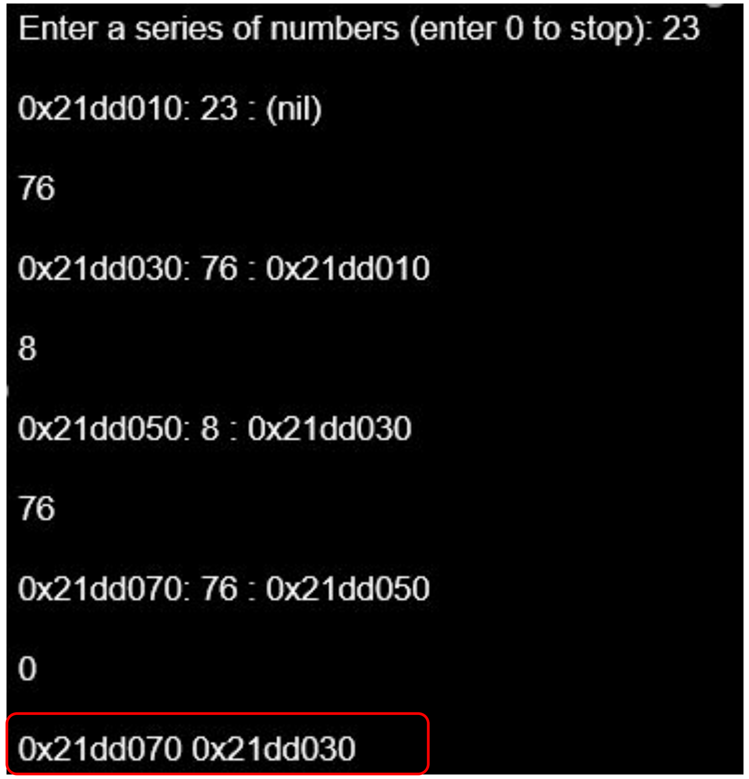


```c
void insertPosition(int pos, int data, struct Node** head)
{
    int size = calcSize(*head);

    //If pos is 0 then should use insertStart method
    //If pos is less than 0 then can't enter at all
    //If pos is greater than size then bufferbound issue
    if(pos < 1 || size < pos)
    {
        printf("Can't insert, %d is not a valid position\n",pos);
    }
    else
    {
        struct Node* temp = *head;
        struct Node* newNode = (struct Node*) malloc(sizeof(struct Node));
        newNode->data = data;
        newNode->next = NULL;

        while(--pos)
        {
            temp=temp->next;
        }
        //(25)->next = 10 as 12->next is 10
        newNode->next= temp->next;
        // (12)->next = 25
        temp->next = newNode;
        //new node added in b/w 12 and 10
    }
}
```

Searching a Linked List

- The following loop searches a linked list for an integer 76:

```
for (temp = first; temp != NULL; temp = temp->next) {
  if (temp->data == 76)
        printf ("%p ", temp);
}
```

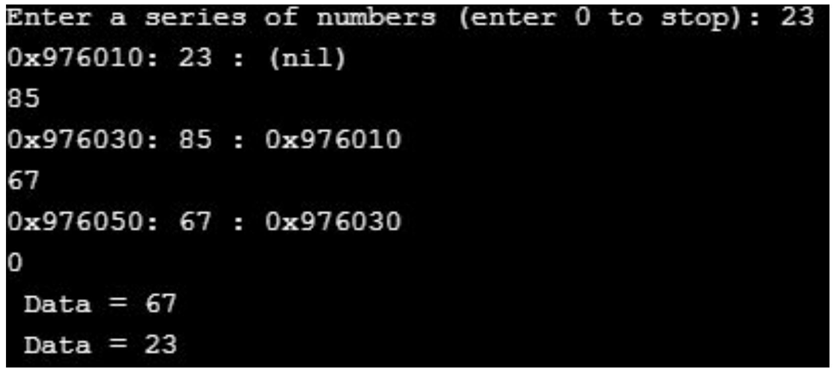


```
Enter a series of numbers (enter 0 to stop): 23

0x21dd010: 23 : (nil)

76

0x21dd030: 76 : 0x21dd010

8

0x21dd050: 8 : 0x21dd030

76

0x21dd070: 76 : 0x21dd050

0

0x21dd070 0x21dd030
```

Deleting from a Linked List

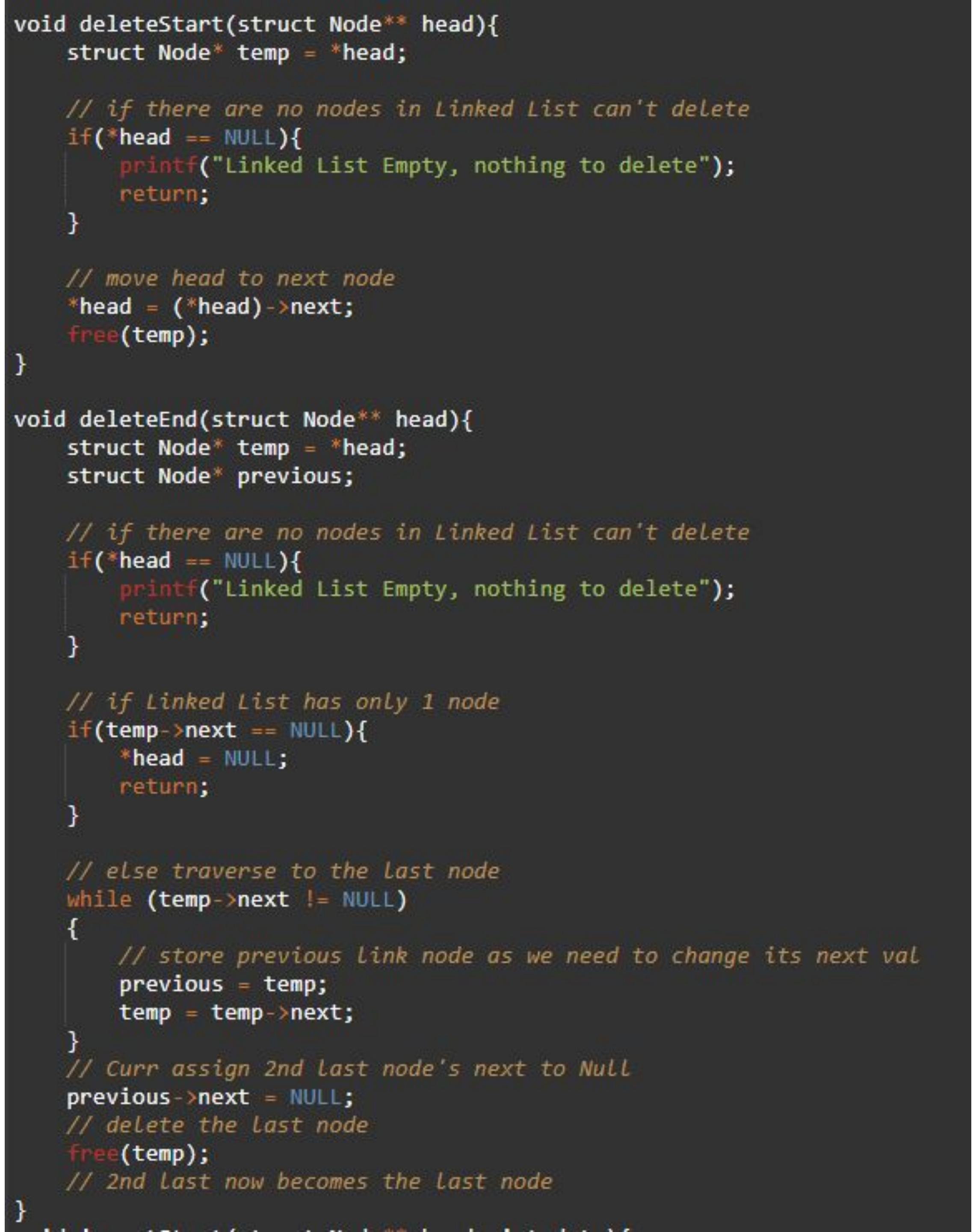
- The following code will delete the first node containing 85 from the list pointed to by first:

```
struct node *p, *q;
for (p = first, q = NULL; p != NULL && p->data != 85; q = p, p = p->next);
if (q == NULL)
   first = first->next; /* 85 is at the beginning */
else
  q->next = p->next; /* 85 is not at the beginning */
free(p);

while(first != NULL)   {
   printf(" Data = %d\n", first->data);
   first = first->next;
}
```



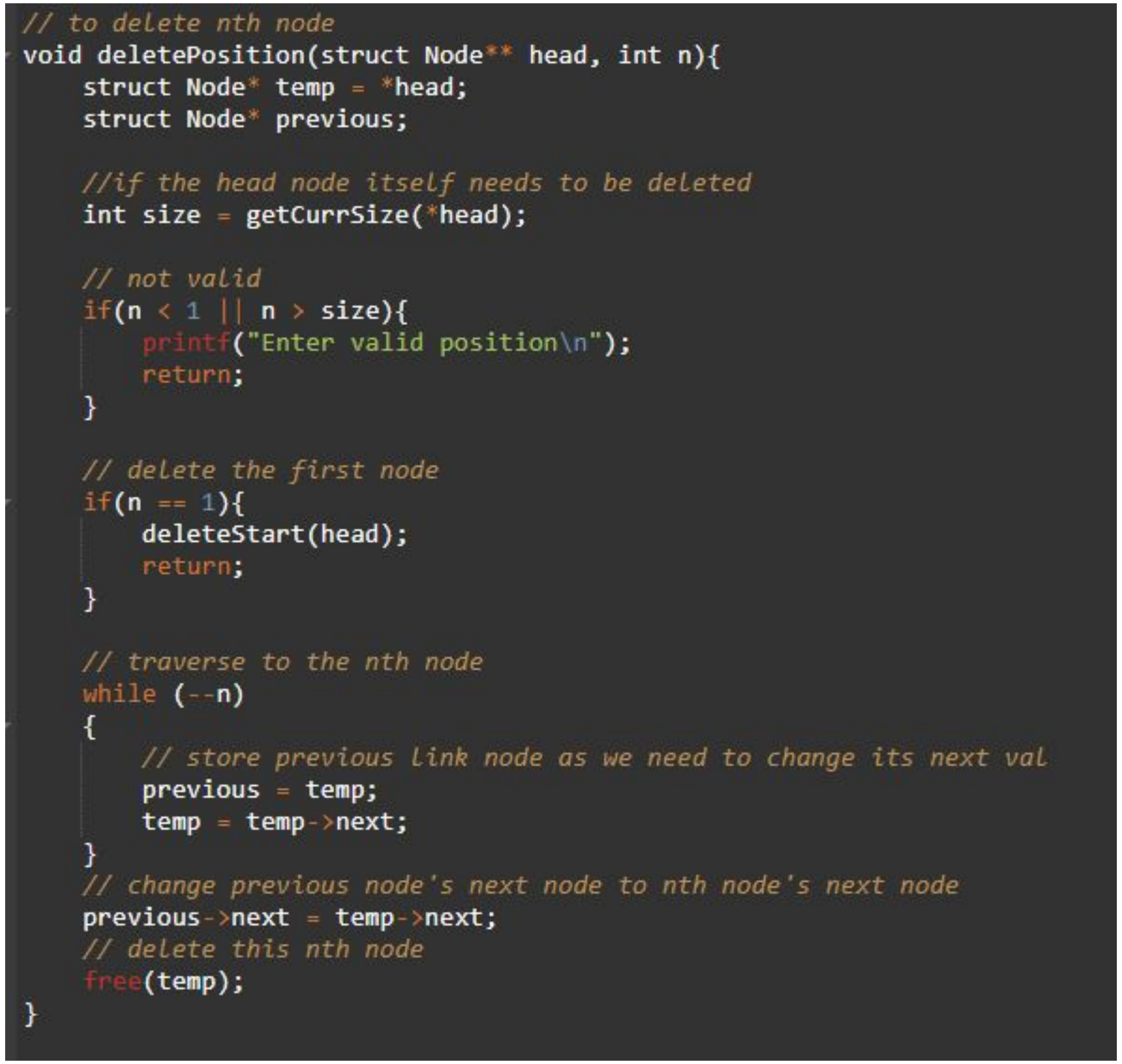
```
Enter a series of numbers (enter 0 to stop): 23
0x976010: 23 : (nil)
85
0x976030: 85 : 0x976010
67
0x976050: 67 : 0x976030
0
 Data = 67
 Data = 23
```

Delete at start & end

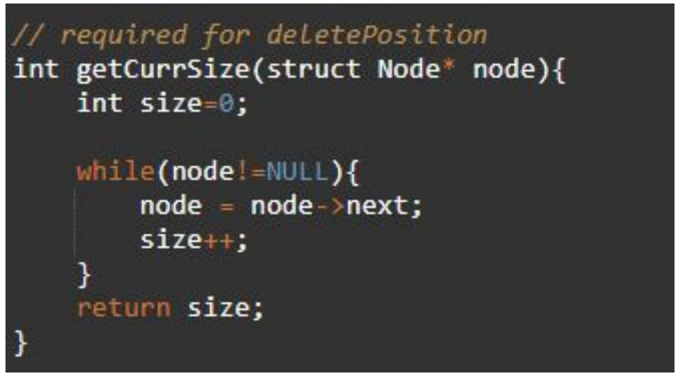


```c
void deleteStart(struct Node** head){
    struct Node* temp = *head;

    // if there are no nodes in Linked List can't delete
    if(*head == NULL){
        printf("Linked List Empty, nothing to delete");
        return;
    }

    // move head to next node
    *head = (*head)->next;
    free(temp);
}

void deleteEnd(struct Node** head){
    struct Node* temp = *head;
    struct Node* previous;

    // if there are no nodes in Linked List can't delete
    if(*head == NULL){
        printf("Linked List Empty, nothing to delete");
        return;
    }

    // if Linked List has only 1 node
    if(temp->next == NULL){
        *head = NULL;
        return;
    }

    // else traverse to the last node
    while (temp->next != NULL)
    {
        // store previous link node as we need to change its next val
        previous = temp;
        temp = temp->next;
    }
    // Curr assign 2nd last node's next to Null
    previous->next = NULL;
    // delete the last node
    free(temp);
    // 2nd last now becomes the last node
}
```

Delete at nth pos



```c
// to delete nth node
void deletePosition(struct Node** head, int n){
    struct Node* temp = *head;
    struct Node* previous;

    //if the head node itself needs to be deleted
    int size = getCurrSize(*head);

    // not valid
    if(n < 1 || n > size){
        printf("Enter valid position\n");
        return;
    }

    // delete the first node
    if(n == 1){
        deleteStart(head);
        return;
    }

    // traverse to the nth node
    while (--n)
    {
        // store previous link node as we need to change its next val
        previous = temp;
        temp = temp->next;
    }
    // change previous node's next node to nth node's next node
    previous->next = temp->next;
    // delete this nth node
    free(temp);
}
```



```c
// required for deletePosition
int getCurrSize(struct Node* node){
    int size=0;

    while(node!=NULL){
        node = node->next;
        size++;
    }
    return size;
}
```