MOHAMMED ABDUL HAMID
1BM19CS202
3-D

Lab 6 (Singly Linked list)

URBAN EDGE 23/11/20 Monday

```
struct node {
        int data;
        struct node * next;
        };
        struct node * head;
void beginsert() {
        strict node * ptr;
        int item;
        ptr =(struct node*) malloc (sizeof (struct node));
        if (Ptr = NULL)
                printf(" In Overflow");
        else {
                printf (" value:\n");
                scanf ("%d", &item);
                ptr->data = item;
                ptr -> next = head;
                head = ptr;
                printf("\nNode inserted");
                }
void lastinsert() {
        struct node *ptr, * temp ;
        int item;
ptr = (struct node*) malloc( size of (struct node));
if (Ptr == NULL)
else { printf (" overflow");
        printf(" Enter item value\n");
        scanf("%d", item);
if (head == NULL) {
        ptr->next = NULL
        head = ptr;
        printf("\n node inserted");
        }
else { temp=head;
        while (temp -> next != NULL) {
                temp = temp ->next;
        }
        temp -> next = ptr;
        ptr -> next = NULL
        }
```

Scanned with CamScanner

```
void randominsert(){
    int i, loc, item
    struct node *ptr, *temp;
    ptr = (struct node *) malloc (sizeof (struct node));
    if (ptr == NULL)
        printf ("overflow");
    else {
        scanf ("%d", &item);
        ptr->data = item;
    printf ("\nEnter location you want to insert");
    scanf ("%d", &loc);
    temp = head;
    for (i = 0; i < loc; i++){
        temp = temp->next;
        if (temp == NULL){
            printf ("\n can't insert");
            return;
        }
    }
    ptr->next = temp->next;
    temp->next = ptr;
    printf ("\n Node inserted");
}

void begin_delete(){
    struct node *ptr;
    if (head == NULL){
        printf ("\n List is Empty\n");
    }
    else {
        ptr = head;
        head = ptr->next;
        free (ptr);
        printf ("\n Node deleted from beginning...\n");
    }
```

Scanned with CamScanner

```
void last_delete () {
        struct node *ptr, *ptr1;
    if (head == NULL) {
            printf (" last is empty");
    else if (head -> next == NULL) {
                head = NULL;
                free (head);
                printf ("\n Only node of list deleted... \n");
    } else {
                ptr = head;
                while (ptr->next != NULL) {
                        ptr1 = ptr;
                        ptr = ptr->next;
                }
                ptr1 ->next = NULL;
                free (ptr);
                printf ("\n Deleted node from last... \n");
    }}

void random_delete() {
        struct node *ptr, *ptr1;
        int loc, i;
    printf ("\n Enter location of node you want to delete.\n");
    scanf ("%d", &loc);
    ptr = head;
    for (i=0 ; i<loc; i++){
            ptr1 = ptr;
            ptr = ptr -> next;
        if (ptr == NULL) {
                printf ("\n Can't delete");
                return;
        } ptr1 -> next = ptr -> next;
        free (ptr);
        printf ("\n Deleted node. %d", loc+1);
    }
```

Scanned with CamScanner

```
void display(){
    struct node *ptr;
    ptr=head;
    if(ptr==NULL){
        printf("Nothing to print");
    }
    else{
        printf("\n printing values ---- \n");
        while(ptr!=NULL){
            printf("\n %d",ptr->data);
            ptr=ptr->next;
        }
    }
```

Scanned with CamScanner