**Day 3 assignment – Data Structures and Algorithms**

1. Node\* insert\_any()(Node \*head, int data, int position)

{

// Complete this method only

// Do not write main function.

Node \* newnode = (Node \*)malloc(sizeof(Node));

newnode->data=data;

Node \*temp=head,\*prev;

if(position==0){

newnode->next=head;

head=newnode;

return head;

}

else{

while(position--){

prev=temp;

temp=temp->next;

}

prev->next=newnode;

newnode->next=temp;

return head;

}

}

1. void delete\_beg() {

    //Checks whether list is empty

    if(head == NULL) {

        return;

    }      else {

        //Checks whether contain only one element

        //If not, head will point to next element in the list and tail will point to new head.

       if(head != tail ) {

            head = head->next;

            tail->next = head;

        }

        //If the list contains only one element

        //then it will remove it and both head and tail will point to null

        else {

            head = tail = NULL;

        }

    }

}

1. **void deleteLastNode()**

**{**

**struct node \*toDelete, \*secondLastNode;**

**if(head == NULL)**

**{**

**printf("List is already empty.");**

**}**

**else**

**{**

**toDelete = head;**

**secondLastNode = head;**

**/\* Traverse to the last node of the list \*/**

**while(toDelete->next != NULL)**

**{**

**secondLastNode = toDelete;**

**toDelete = toDelete->next;**

**}**

**if(toDelete == head)**

**{**

**head = NULL;**

**}**

**else**

**{**

**/\* Disconnect link of second last node with last node \*/**

**secondLastNode->next = NULL;**

**}**

**/\* Delete the last node \*/**

**free(toDelete);**

**printf("SUCCESSFULLY DELETED LAST NODE OF LIST\n");**

**}**

**}**