Name: Sadat Zubin Aftab Shah Roll number: 20CSE1030

1) Concatenate two given list into one big list.

node *concatenate (node *head1, node *head2);

```c
#include<stdio.h>
#include<stdlib.h>
#define take(n) scanf("%d",&n);
#define rep(i,a,b) for(int i=a;i<b;i++)

typedef struct nod{
    int data;
    struct nod *next;
}node;

node* takeinput(){
    printf("Enter the number of elements in the linkedlist\n");
    int n;
    take(n);
    node *temp,*head=NULL;
    rep(i,0,n){
        node *current=(node*) malloc(sizeof(node));
        int value;
        printf("Enter the value of element no %d : ",i+1);
        take(value);
        current->data=value;
        current->next=NULL;
        if(i==0) head=current;
        else  temp->next=current;
        temp=current;
    }
    return head;
}

void printlinkedlist(node *head){

    node *temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}

node *concatenate(node *node1,node *node2){
    node *head=NULL;
    node *temp,*prev;
    temp=node1;
```

```
    while(temp!=NULL){
        node *current=(node *) malloc(sizeof(node));
        current->data=temp->data;
        current->next=NULL;
        if(head==NULL) head=current;
        else prev->next=current;
        temp=temp->next;
        prev=current;
    }
    temp=node2;

    while(temp!=NULL){
        node *current=(node *) malloc(sizeof(node));
        current->data=temp->data;
        current->next=NULL;
        if(head==NULL) head=current;
        else prev->next=current;
        temp=temp->next;
        prev=current;
    }
    return head;
}
void delete_linkedlist(node **head){
    node *temp=*head;
    while(*head!=NULL){
        temp=*head;
        *head=(*head)->next;
        free(temp);
    }
}
int main()
{
    printf("\n");
    node *node1=takeinput();
    node *node2=takeinput();
    node *node3=concatenate(node1,node2);
    printf("\n");

    printf("First Linkedlist : ");
    printlinkedlist(node1);

    printf("Second Linkedlist : ");
    printlinkedlist(node2);

    printf("Concatenated Linkedlist : ");
    printlinkedlist(node3);

    delete_linkedlist(&node1);
    delete_linkedlist(&node2);
    delete_linkedlist(&node3);
return 0;
}
```

TERMINAL

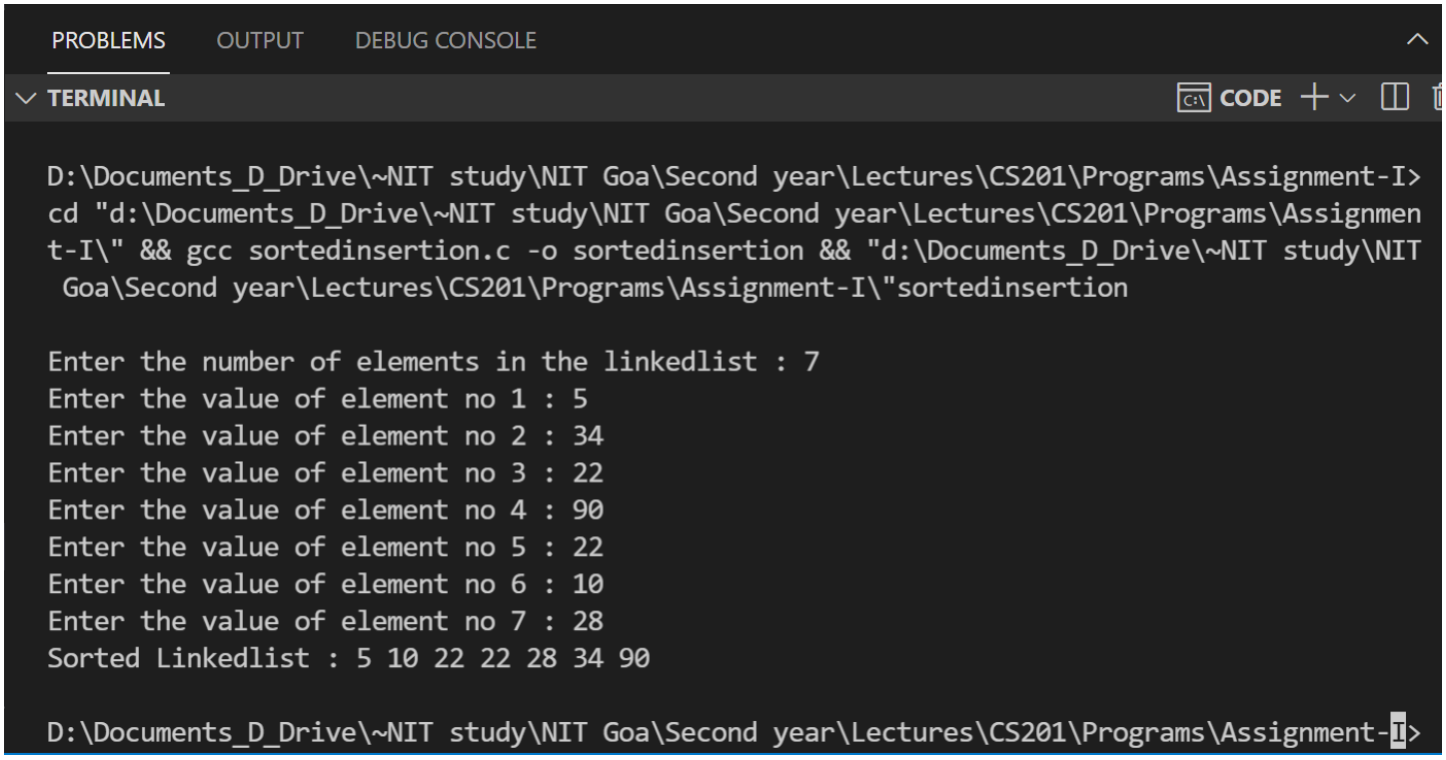
```
ond year\Lectures\CS201\Programs\Assignment-I\"concatenate

Enter the number of elements in the linkedlist
6
Enter the value of element no 1 : 21
Enter the value of element no 2 : 22
Enter the value of element no 3 : 23
Enter the value of element no 4 : 24
Enter the value of element no 5 : 25
Enter the value of element no 6 : 26
Enter the number of elements in the linkedlist
4
Enter the value of element no 1 : 31
Enter the value of element no 2 : 32
Enter the value of element no 3 : 33
Enter the value of element no 4 : 34

First Linkedlist : 21 22 23 24 25 26
Second Linkedlist : 31 32 33 34
Concatenated Linkedlist : 21 22 23 24 25 26 31 32 33 34

D:\Documents_D_Drive\~NIT study\NIT Goa\Second year\Lectures\CS201\Programs\Assignment-I>
```

2) Insert an element in a linked list in sorted order. the function will be called for every element to be inserted.

void insert_sorted (node **head, node *element);

```c
#include<stdio.h>
#include<stdlib.h>
#define take(n) scanf("%d",&n);
#define rep(i,a,b) for(int i=a;i<b;i++)

typedef struct nod{
    int data;
    struct nod *next;
}node;

void insert_sorted(node **head,node *element){
    node *temp=*head,*prev;
    while(temp!=NULL && element->data>temp->data){
        prev=temp;
        temp=temp->next;
    }
    if(temp!=*head){
    prev->next=element;
    element->next=temp;
    }
    else {
        element->next=*head;
        *head=element;
    }
}

node *takesortedinput(){
    node *head=NULL,*temp;
    printf("Enter the number of elements in the linkedlist : ");
    int n;
    take(n);
    rep(i,0,n){
        node *element=(node *) malloc(sizeof(node));
        printf("Enter the value of element no %d : ",i+1);
        int value;
        take(value);
        element->data=value;
        element->next=NULL;
        if(head==NULL) head=element;
        else insert_sorted(&head,element);
    }
    return head;
}

void printlinkedlist(node *head){
    printf("Sorted Linkedlist : ");
    node *temp=head;
    while(temp!=NULL){
```

```
        printf("%d ",temp->data);
        temp=temp->next;
    }
    printf("\n");
}
void delete_linkedlist(node **head){
    node *temp=*head;
    while(*head!=NULL){
        temp=*head;
        *head=(*head)->next;
        free(temp);
    }
}
int main()
{
    printf("\n");
    node *node1=takesortedinput();
    printlinkedlist(node1);
    delete_linkedlist(&node1);
return 0;
}
```



3) Always insert elements at one end, and delete elements from the other end (first-in first-out Queue).

void insert_q (node **head, node *element)

node *delete_q (node **head) /* Return the deleted node */

```c
#include <stdio.h>
#include <stdlib.h>
#define take(n) scanf("%d", &n);
#define rep(i, a, b) for (int i = a; i < b; i++)

typedef struct nod{
    int data;
    struct nod *next;
} node;

void insert_q(node **head, node *current)
{
    if (*head == NULL){
        *head = current;
        return;
    }
    node *temp = *head, *prev = NULL;
    while (temp != NULL){
        prev = temp;
        temp = temp->next;
    }
    prev->next = current;
}

void takeinput(node **head)
{
    printf("Enter the value of the element : ");
    int value;
    take(value);
    node *current = (node *)malloc(sizeof(node));
    current->data = value;
    current->next = NULL;
    insert_q(head, current);
}

node *delete_q(node **head)
{
    if (*head == NULL) return NULL;
    else{
        node *temp = *head;
        *head = (*head)->next;
        temp->next = NULL;
        return temp;
    }
}

void printlinkedlist(node *head)
{
```

```
    if (head == NULL) printf("Linkedlist is empty\n");
    else
    {
        printf("Current linkedlist is: ");
        node *temp = head;
        while (temp != NULL)
        {
            printf("%d ", temp->data);
            temp = temp->next;
        }
        printf("\n");
    }
}

void delete_linkedlist(node **head)
{
    node *temp = *head;
    while (*head != NULL)
    {
        temp = *head;
        *head = (*head)->next;
        free(temp);
    }
}

int main()
{
    node *queue;
    node *deleted;
    int choice;
    do{
        printf("Press 1 to input element inside queue\n");
        printf("Press 2 to delete the element from the queue\n");
        printf("Press 0 to exit\n");
        take(choice);
        switch (choice){
        case 1:
            takeinput(&queue);
            printlinkedlist(queue);
            break;
        case 2:
            deleted = delete_q(&queue);
            if (deleted != NULL){
                printf("value deleted is:%d\n", deleted->data);
                free(deleted);
            }
            printlinkedlist(queue);
        default:
            break;
        }
    } while (choice);
    delete_linkedlist(&queue);
    return 0;
}
```

```
                                                                                c
d "d:\Documents_D_Drive\~NIT study\NIT Goa\Second year\Lectures\CS201\Programs\Assignment
-I\" && gcc queue.c -o queue && "d:\Documents_D_Drive\~NIT study\NIT Goa\Second year\Lect
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
2
Linkedlist is empty
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
1
Enter the value of the element : 23
Current linkedlist is: 23
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
1
Enter the value of the element : 56
Current linkedlist is: 23 56
Press 1 to input element inside queue
Press 2 to delete the element from the queue
```

```
Press 0 to exit
1
Enter the value of the element : 78
Current linkedlist is: 23 56 78
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
2
value deleted is:23
Current linkedlist is: 56 78
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
2
value deleted is:56
Current linkedlist is: 78
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
2
value deleted is:78
Linkedlist is empty
```

```
Press 1 to input element inside queue
Press 2 to delete the element from the queue
Press 0 to exit
0


D:\Documents_D_Drive\~NIT study\NIT Goa\Second year\Lectures\CS201\Programs\Assignment-I>
```

4) Assume two polynomials are represented by a linked list. Write a function that adds two polynomials. (Add the coefficients of same variable powers).

Void poly_add(node *list1, node *list2, node *list) /* function adding two polynomial numbers

```c
#include<stdio.h>
#include<stdlib.h>
#define take(n) scanf("%d",&n);
#define rep(i,a,b) for(int i=a;i<b;i++)

typedef struct nod{
    int degree;
    int coeff;
    struct nod* next;
}node;

void insert_node(node **head,node *current){
    node *temp=*head,*prev;

    while(temp!=NULL && temp->degree>=current->degree){
        prev=temp;
        temp=temp->next;
    }
    if(temp!=*head){
    prev->next=current;
    current->next=temp;
    }
    else{
        current->next=*head;
        *head=current;
    }

}

node *takeinput(){
    node *head=NULL;
    printf("Enter the number of terms in the polynomial: ");
    int n;
    take(n);
    rep(i,0,n){
        printf("Degree: ");
        int deg;
        take(deg);
        node *current=(node *) malloc(sizeof(node));
        printf("Coefficient of degree %d: ",deg);
        int cof;
        take(cof);
        current->next=NULL;
        current->degree=deg;
        current->coeff=cof;
        if(head==NULL) head=current;
        else insert_node(&head,current);
    }
```

```
    return head;
}

void printpoynomial(node *head){
    node *temp=head;
    int first=1;
    while(temp!=NULL){
        if(first==1) first=0;
        else printf(" + ");
        if(temp->degree!=0) printf("%dx^%d",temp->coeff,temp->degree);
        else printf("%d",temp->coeff);
        temp=temp->next;
    }
    printf("\n");
}

void poly_add(node *poly1,node *poly2,node **poly3){
    *poly3=NULL;
    node *temp1=poly1;
    node *temp2=poly2;
    node *temp;
    while(temp1!=NULL && temp2!=NULL){
        node *current=(node*) malloc(sizeof(node));
        if(temp1->degree>temp2->degree){
            current->coeff=temp1->coeff;
            current->degree=temp1->degree;
            temp1=temp1->next;
        }
        else if(temp1->degree<temp2->degree){
            current->coeff=temp2->coeff;
            current->degree=temp2->degree;
            temp2=temp2->next;
        }
        else {
            current->coeff=temp1->coeff+temp2->coeff;
            current->degree=temp1->degree;
            temp1=temp1->next;
            temp2=temp2->next;
        }
        current->next=NULL;
        if(*poly3==NULL) {
            *poly3=current;
            temp=current;
        }
        else {
            temp->next=current;
            temp=temp->next;
        }
    }
    while(temp1!=NULL){
            node *current=(node*) malloc(sizeof(node));
            current->coeff=temp1->coeff;
            current->degree=temp1->degree;
            temp1=temp1->next;
```

```
        if(*poly3==NULL) {
            *poly3=current;
            temp=current;
        }
        else {
            temp->next=current;
            temp=temp->next;
        }
            current->next=NULL;
    }
    while(temp2!=NULL){
            node *current=(node*) malloc(sizeof(node));
            current->coeff=temp2->coeff;
            current->degree=temp2->degree;
            temp2=temp2->next;
        if(*poly3==NULL) {
            *poly3=current;
            temp=current;
        }
        else {
            temp->next=current;
            temp=temp->next;
        }
            current->next=NULL;
    }
}
void delete_poly(node **head){
    node *temp=*head;
    while(*head!=NULL){
        temp=*head;
        *head=(*head)->next;
        free(temp);
    }
}
int main()
{
    printf("\n");
    node *poly1=takeinput();
    node *poly2=takeinput();
    node *poly3;
    poly_add(poly1,poly2,&poly3);

    printf("First Polynomial : ");
    printpoynomial(poly1);

    printf("Second Polynomial : ");
    printpoynomial(poly2);

    printf("Sum of above two Polynomials : ");
    printpoynomial(poly3);

    delete_poly(&poly1);
    delete_poly(&poly2);
    delete_poly(&poly3);
```

```
return 0;
}
```

```
Enter the number of terms in the polynomial: 5
Degree: 0
Coefficient of degree 0: 5
Degree: 5
Coefficient of degree 5: 2
Degree: 8
Coefficient of degree 8: 21
Degree: 3
Coefficient of degree 3: 32
Degree: 9
Coefficient of degree 9: 44
Enter the number of terms in the polynomial: 3
Degree: 2
Coefficient of degree 2: 4
Degree: 5
Coefficient of degree 5: 32
Degree: 0
Coefficient of degree 0: 1
First Polynomial : 44x^9 + 21x^8 + 2x^5 + 32x^3 + 5
Second Polynomial : 32x^5 + 4x^2 + 1
Sum of above two Polynomials : 44x^9 + 21x^8 + 34x^5 + 32x^3 + 4x^2 + 6
```