① . Write a program to insert and delete a node in linked list .

⇒ Program to insert and delete a node -

```
#include<stdio.h>
#include<stdlib.h>
struct node {
int data;
Struct node *next;
};
typedef struct node *Node;
Node *start, *temp, *ptr, *N;
void create_and_insert();
void delete_list();
void display();
int main()
{
  Stust = NULL;
  create_and_ insert();
  display();
  delete_list();
  return 0;
}
void create_and_ insert()
{
 int choice;
 do {
    printf("Enter the data");
    N = (Node *) malloc (sizeof (Node));
    scanf("%d", & N→data);
    if (start == NULL)
    {
     start = temp = N;
    }
```

```
else {
 temp->next = N;
 temp=N;
 }
 printf("Press 1 to insert data");
 scanf("%d", &choice);
}while(choice==1);
temp->next = NULL;
}

void delete_list() {
 int ch;
 printf("Press 2 to delete data");
 scanf("%d", &ch);
 if(ch==2) {
    temp = start;
    ptr = start;
  while(temp->next != NULL)
  { ptr= temp;
    temp = temp->next;
   }
   ptr->next =NULL;
  free(temp);
  display();
  }
}
 void display() {
  temp=start;
  printf("Your list is:");
  while(temp!=NULL) {
  printf("%d\t", temp->data);
  temp = temp->next;
  }
 }
```

64 lines

Output :

```
Enter the data : 10
Press 1 to insert data : 1
Enter the data : 20
Press 1 to insert data : 1
Enter the data : 30
Press 1 to insert data : 2
Your list is : 10 20 30
Press 2 to delete data : 2
Your list is : 10 20
```

② Write a program to add two polynomials using linked list.

```
#include <stdio.h>
#include <stdlib.h>

typedef struct link
{
  int coff;
  int pow;
  struct link *next;
}my_poly;

void my_create_poly (my_poly **);
void my_show_poly (my_poly *);
void my_add_poly (my_poly **, my_poly *, my_poly*);

int main(void)
{
 int ch;
 do{
 my_poly *poly1, *poly2, *poly3;
 printf("\n Create 1st expression");
 my_create_poly (&poly1);
 printf("\nstored the 1st expression");
 my_show_poly(poly1);
 printf ("\n Create 2nd expression");
 my_create_poly(& poly2);
 printf ("\n stored the 2nd expression");
 my_show_poly(poly2);
 my_add_poly (& poly3, poly1, poly2);
 my_show_poly (poly3);
 printf ("\n Add two more expression (Y=1/N=0:");
 scanf ("%d", &ch);
 }while (ch);
 return 0;
}
```

```
void my_create_poly(my_poly **node)
{
  int flag;
  int coeff, pow;
  my_poly *tmp_node;
  tmp_node = (my_poly *)malloc (sizeof(my_poly));
  *node = tmp_node;
  do {
    printf("\n Enter coeff :");
    scanf ("%d", &coeff);
   temp_node → coeff = coeff;
   printf ("\n Enter Pow :");
   scanf ("%d", &pow);
   tmp_node → pow = pow;
   tmp_node → next = NULL;
  printf ("\n continue adding more terms to the
          polynomial list ? (Y=1/N=0) : ");
  scanf ("%d", &flag);
  if (flag) {
  tmp_node → next = (my_poly *)malloc(sizeof(my_poly));
   tmp_node = tmp_node → next;
   temp_node → next = NULL;
  }
}while (flag);
}
 void my_show_poly (my_poly *node)
 {
 printf ("\n The polynomial expression is : \n");
 while (node != null)
  {printf ("%.dx^%d", node → coeff, node → pow);
  node = node → next;
  if (node != NULL)
     printf(" + ");
  }
}
```

```
void my_add_poly(my_poly **result, my_poly *poly1,
                 my_poly *poly2)
{
 my_poly *tmp_node;
 tmp_node = (my_poly *)malloc(sizeof(my_poly));
 tmp_node->next = NULL;
 *result = tmp_node;
 while (poly1 && poly2)
 {
  if (poly1->pow > poly2->pow)
  {
   tmp_node->pow = poly1->pow;
   tmp_node->coeff = poly1->coeff;
   poly1 = poly1->next;
  }
  else if (poly1->pow < poly2->pow)
  {
   tmp_node->pow = poly2->pow;
   tmp_node->coeff = poly2->coeff;
   poly2 = poly2->next;
  }
  else {
   tmp_node->pow = poly1->pow;
   tmp_node->coeff = poly1->coeff + poly2->coeff;
   poly1 = poly1->next;
   poly2 = poly2->next;
  }
  if (poly1 && poly2)
  {
   tmp_node->next = (my_poly *)malloc(sizeof(my_poly));
   tmp_node = tmp_node->next;
   tmp_node->next = NULL;
  }
 }
```

```
while (poly1 || poly2)
{
tmp_node->next=(my_poly*)malloc(sizeof(my_poly));
tmp_node= tmp_node->next;
tmp_node->next = NULL;
if (poly1)
{
 tmp_node->pow = poly1->pow;
 tmp_node->coeff = poly1->coeff;
 poly1 = poly1->next;
}
 if (poly2)
 {
 tmp_node->pow = poly2->pow;
 tmp_node->coeff=poly2->coeff;
 poly2 = poly2->next;
 }
}
 printf ("\n Addition complete");
}
```

Output -

```
Create 1st expression
Enter Coeff: 7
Enter pow: 3
Continue adding more terms to the polynomical
list?(Y=1/N=0): 1
Enter coeff: 6
Enter pow: 2
Continue adding more terms to the polynomial
list ?(Y=1/N=0): 0
Stored the 1st expression
The polynomial expression is:
```

7x^3 + 6x^2

Create 2nd Expression

Enter coeff: 3

Enter pow: 3

Continue adding more terms to the polynomials list ? (Y=1/N=0): 1

Enter coeff: 4

Enter pow: 2

Continue adding more terms to the polynomial list ? (Y=1/N=0): 0

Stored the 2nd expression

The polynomial expression is:

3x^3 + 4x^2

Addition Complete

The polynomial expression is:

10x^3 + 10x^2

Add two more expression ? (Y=1/N=0): 0

③. Write a program to insert & delete a node in a circular linked list.

⇒ Insertion At Beginning

```
#include <stdio.h>
#include<stdlib.h>
struct node {
 int info;
 struct node *next;
 };
struct node *last =NULL;
void insertAtFront(int data)
{
 struct node *temp;
 temp = (struct node *)malloc (sizeof(struct node));
 if (last==NULL) {
 temp→info =data;
 temp→next= temp;
 last = temp;
 }
 else {
 temp→info=data;
 temp→next= last→next;
 last→next = temp;
 }
}
 void display()
 {
  if (last==NULL)
     printf("\n List is empty\n");
   else {
     struct node *temp;
     temp=last→next;
     do
     {
     printf ("\n Data=%d", temp→info);
     temp= temp→next;
     }while (temp!=last→next);
  }}
```

```
int main()
{
insertAtFront(10);
insertAtFront(20);
insertAtFront(30);
display();
return 0;
}
```

Output :

```
Data : 30
Data : 20
Data : 10
```

Insert at end-

```
#include<stdio.h>
#include<stdlib.h>
struct node {
   int info;
   struct node *next;
};
struct node* last = NULL;
void addatlast (int data)
{
 struct node *temp;
 temp=(struct node *)malloc (sizeof (struct node));
if ( last == NULL)
{
   temp→info= data;
   temp→next=temp;
   last=temp;
}
else
{
 temp→info = data;
 temp→next = last→next;
last→next = temp;
last= temp;
}}
```

```
void display ( )
{
 if (last == NULL)
   printf ("\n List is empty \n");
 else {
   struct node *temp;
   temp = last -> next;
   do {
     printf ("\n Data = %d", temp -> info);
     temp = temp -> next;
   } while (temp != last -> next);
 }
}

int main ( )
{
 addatlast (10);
 addatlast (20);
 addatlast (30);
 display ( );
 return 0;
}
```

Output :

Data : 10
Data : 20
Data : 30

Deletion at any point

```
#include<stdio.h>
#include<stdlib.h>
struct node {
   int info;
   struct node *next;
};
struct node *last = NULL;
void addatlast()
{
 int data;
 struct node *temp;
temp = (struct node *)malloc(sizeof(struct node));
printf("\nEnter data to be inserted:\n");
scanf("%d", &data);
if (last == NULL) {
   temp->info = data;
   temp->next = temp;
   last = temp;
 }
 else {
   temp->info = data;
   temp->next = last->next;
   last->next = temp;
   last = temp;
 }
}

 void deleteatindex()
 {
   int pos, i = 1;
   struct node *temp, *position;
   temp = last->next;
   if (last == NULL)
     printf("\nList is empty.\n");
```


P.T.O →

```
else {
  printf("\nEnter index :");
  scanf("%d",&pos);
  while(i<=pos-1){
    temp= temp->next;
    i++;
  }
  position =temp->next;
  temp->next =position ->next;
  free(position);
  }
}
void display()
{
if(last==NULL)
   printf("\n List is empty\n");
else {
    struct node *temp;
    temp=last->next;
    do{
       printf("\nData=%d",temp->info);
       temp=temp->next;
    }while(temp!=last->next);
  }
}

int main()
{
 addatlast();
 addat last();
 addatlast();
 delete atindex();
 display();
 return 0;
}
```

Output -

```
Enter data to be inserted:
10

Enter data to be inserted:
20

Enter data to be inserted:
30

Enter index: 2
Data = 10
Data = 30
```