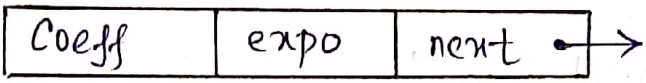
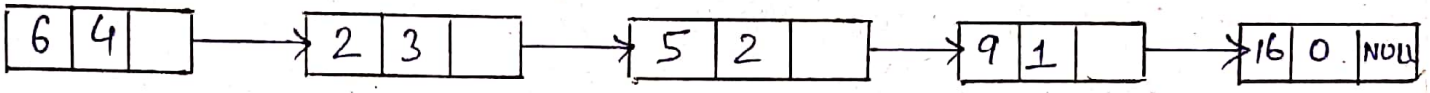
Linked - Lists And Polynomials

We can represent a polynomial like $6x^4 + 2x^3 + 5x^2 + 9x + 16$ with the help of linked list.

In the linked representation of polynomials, each node should consists of three elements ie coefficient, exponent and a link to the next term.



The linked-list representation of above polynomial is



C program to create and display a polynomial using Linked list.

```
#include<stdio.h>
#include <alloc.h>
#include <conio.h>

struct node
{
    int coeff;
    int expo;
    struct node * next;
};

struct node * start = NULL;

void create();
void display();
```

```
void main()
{
int ch;
clrscr();
while (1)
{
printf("\n Representation of polynomial using Linked List");
printf("1. Create \n");
printf("2. Display \n");
printf("3. Quit \n");
printf(" Enter your choice: ");
scanf("%d", &ch);
switch (ch)
{
case 1: create();
        break;
case 2: display();
        break;
case 3: exit(1);
default : printf(" Invalid choice \n");
} /* End of switch */
} /* End of while */
} /* End of main */
```

```
void create()
{
struct node * temp;
temp= (struct node *) malloc (sizeof (struct node));
printf ("Enter the term coefficient and exponent : ");
scanf ("%d %d", &temp -> coeff, &temp -> expo);
temp -> next = NULL;
if (start == NULL)
   {
    start = temp;
   }
else
   {
    struct node *p;
    p= start;
  while (p -> next != NULL)
    {
     p= p -> next;
    }
     p -> next = temp;
   }
printf ("New node has been inserted \n");
}
    void display()
   {
    struct node * temp;
    temp = start ;
    if (temp == NULL)
     {
     printf ("No nodes in the list \n");
     }
```

```
else{
    while (temp != NULL)
    {
    printf ("coefficient : %d \n", temp -> coeff);
    printf ("exponent : %d \n", temp -> expo);
    temp = temp -> next;
    }
  }
}
```

Simple Algorithm for addition of two polynomial

(1) Read coefficient and exponent of first polynomial.

(2) Read coefficient and exponent of second polynomial.

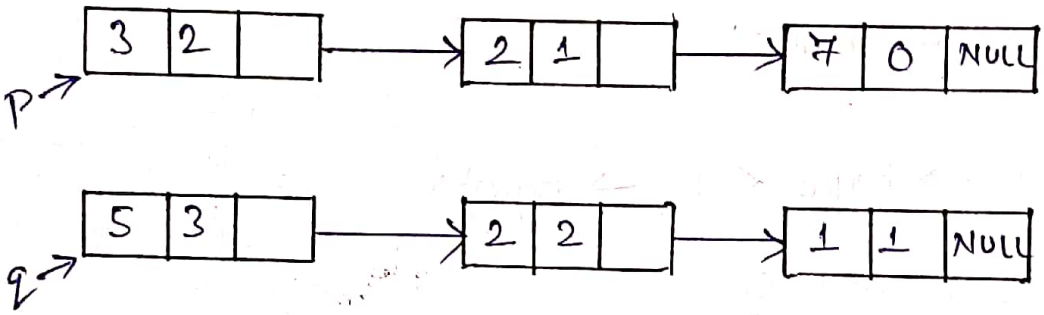
(3) Set two pointers 'p' and 'q' to traverse two polynomials respectively.

(4) Compare the exponent of two polynomials starting from first node.

example:

$$p = 3x^2 + 2x + 7$$

$$q = 5x^3 + 2x^2 + x$$



(i) if both exponents are equal then add coefficient and store in result linked list and move p and q to point next node.

(ii) if exponent of p < exponent of q, then add terms of q in result and move q to point next node.

(iii) if exponent of p > exponent of q, then add terms of p in result and move p to point next node.

fig. Resultant Polynomial

```
// Function Adding two polynomial numbers
void polyadd(struct Node *poly1, struct Node *poly2, struct Node *poly)
{
while(poly1->next && poly2->next)
        {
                // If power of 1st polynomial is greater then 2nd, then store 1st as it is and move
its pointer
                if(poly1->pow > poly2->pow)
                {
                        poly->pow = poly1->pow;
```

```
                poly->coeff = poly1->coeff;
                poly1 = poly1->next;
        }

        // If power of 2nd polynomial is greater then 1st, then store 2nd as it is  and move
its pointer
        else if(poly1->pow < poly2->pow)
        {
                poly->pow = poly2->pow;
                poly->coeff = poly2->coeff;
                poly2 = poly2->next;
        }

        // If power of both polynomial numbers is same then add their coefficients
        else
        {
                poly->pow = poly1->pow;
                poly->coeff = poly1->coeff+poly2->coeff;
                poly1 = poly1->next;
                poly2 = poly2->next;
        }

        // Dynamically create new node
        poly->next = (struct Node *)malloc(sizeof(struct Node));
        poly = poly->next;
        poly->next = NULL;
    }
while(poly1->next || poly2->next)
    {
        if(poly1->next)
        {
                poly->pow = poly1->pow;
                poly->coeff = poly1->coeff;
                poly1 = poly1->next;
        }
        if(poly2->next)
        {
                poly->pow = poly2->pow;
                poly->coeff = poly2->coeff;
                poly2 = poly2->next;
        }
        poly->next = (struct Node *)malloc(sizeof(struct Node));
        poly = poly->next;
        poly->next = NULL;
    }
}
```