SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)

DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJ 19 ITL 302 **DATE:** DJ 19 ITL 302

COURSE NAME: Data Structures & Algorithms Lab **CLASS:** S.Y. B.Tech(Sem III)

EXPERIMENT NO.1

CO/LO: Solve the problem using Singly linked List

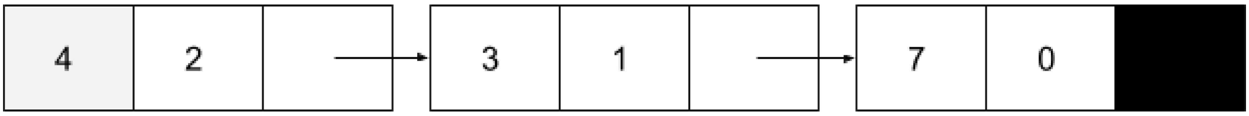
AIM / OBJECTIVE: Write a program to perform polynomial operations (addition, subtraction, multiplication) using linked List.

Each polynomial should be represented as a list with linked list implementation. The first node in the list represents the first Each polynomial should be represented as a list with linked list implementation. The first node in the list represents the first term in the polynomial; the 2nd node represents the 2nd term and so on. The structure of a node is as follows.

Structure of a Node:

```
struct Node
 {
int coefficient;
int power;
struct Node link;
};
```

Ex. 4x^2 + 3x+ 7



DESCRIPTION OF EXPERIMENT:

To Perform following Basic Operations:

1. Create a polynomial
2. Printing a polynomial

3. Addition of two polynomials
4. Subtraction of two polynomials
5. Multiplication of two polynomials

INPUT DATA / DATASET:

FOR ADDITION

POLYNOMIAL 1:3X^2+1X^1

POLYNOMIAL 2: 1X^1

FOR SUBTRACTION

POLYNOMIAL 1:3X^2+1X^1

POLYNOMIAL 2:1X^1

FOR MULTIPLICATION

POLYNOMIAL 1:3X^2

POLYNOMIAL 2:2X^1

PROCEDURE / ALGORITHM:

Create a polynomial:

STEP 1:START

STEP 2:Input the coefficient and power of the variable in the decreasing order of the power.

STEP 3:Create Node (coefficient, power, polynomial) = new node

STEP 4:Insert (new node, polynomial)

STEP 5:End.

Creating a node:

STEP 1:START

```
STEP 2:Allocate memory to temporary node namely ‘x’
STEP 3:x(co-efficient)=polynomial(co-efficient)
       x(power)=polynomial(power)
       x(link)= NULL
STEP 4:RETURN x
STEP 5:END
```

Inserting a new node:

```
STEP 1:START
STEP 2:If head == null then head =new node
STEP 3:Else
        ptr= head
        while(ptr(link)!=null)

        ptr=ptr(link)
        End while statement
        ptr(link)= new node
        End If
STEP 4:END
```

Printing a polynomial:

```
STEP 1:BEGIN
STEP 2: ptr= head
STEP 3:WHILE (ptr (link) ! = NULL)
       Display ptr (coefficient) + "x^" +ptr (power) + "+"
       ptr=ptr (link)
      END WHILE
STEP 4: END
```

Addition of two polynomials:

```
STEP 1:BEGIN.
STEP 2:WHILE (P1(link) != NULL AND P2(link)!=NULL)
        Follow Steps 3 to step 8.
STEP 3: IF (P1(power) == P2 (power)) THEN
        POLY (POWER) = P1 (POWER)
        POLY(COEFF) =P1(COEFF) + P2(COEFF)
        P1 = P1(link)
        P2 = P2 (link)
STEP 4:ELSE IF P1(POWER) > P2 (POWER) THEN
        POLY(POWER). = P1(POWER).
STEP 5: ELSE
        POLY (POWER) =P2 (POWER)
        POLY (COEFF) = P2 (COEFF)
        P2 = P2 (link)
STEP 6:Create Node (coeff, power, poly)=newnode
STEP 7:NewNode =Poly (link)
STEP 8:POLY (link)=NULL
STEP 9: WHILE (P1(link) ==NULL AND P2 (link)!= NULL) OR
         (P1(link)!= NULL AND P2(link) ==NULL)
          Follow Steps 10 to 14.
STEP 10: IF (P1 (link) !=NULL) THEN
            POLY(POWER) = P1 (POWER)
            poly(coeff)=p1(coeff)
            p1=p1(link)
STEP 11: IF (p2(link) != NULL) THEN
            POLY(POWER) = P2(POWER)
            poly(coeff)=p2(coeff)
          p2=p2(link)
STEP 12: New Node = Create Node (coeff, power, poly)
STEP 13: New Node = poly (link)
STEP 14: poly (link) = Null
STEP 15: END
```

Subtraction of two polynomials:

```
STEP 1:BEGIN
STEP 2:WHILE (P1(link) != NULL AND P2 (link) !=NULL
             Follow        Steps 3 to 8
STEP 3:IF (P1(POWER) == P2 (POWER))
       THEN
       POLY(POWER) = P1 (POWER).
       POLY (COEPP) = P1 (COEFF) - P2 (COEFF).
        P1 = P1 (link)
       P2 = P2(link).

STEP 4: ELSE IF P1(POWER) > P2 (POWER)
       THEN
       POLY (POWER) = P1 (POWER)
       POLY (COEFF) = P1(COEFF)
       $P_1$ = P1 (link)
STEP 5:ELSE
       POLY (POWER) = P2 (POWER)
       POLY (COEFF) - P2 (COEFF)
       P2 = P2 (link)
STEP 6: NewNode = Create Node (coeff, power, poly)
STEP 7: NewNode=poly(link)
STEP 8: POLY(link) = NULL
STEP 9:WHILE(P1(link) = = NULL AND P2 (link) != NULL) OR
         (P1 (link) != NULL AND P2 (link) = = NULL)
         Follow steps 10 to 14
STEP 10:IF (P1 (link) != NULL)
        THEN POLY (POWER) = P1 (POWER)
        POLY (COEFF) = P1 (COEFF)
        P1 = P1 (link)
STEP 11: IF( P2(link) != NULL) THEN
        POLY (POWER) = P2 (POWER)
        POLY (COFFF) = = P2(COEFF)
        P2 = P2 (link)
STEP 12:New Node = Create Node (COEFF, POWER, POLY)
STEP 13:New Node = poly (link)
STEP 14 :poly (link)=NULL
STEP 15:END
```

Multiplication of two polynomials:

```
STEP 1 :BEGIN
STEP 2:WHILE (P1(link) != NULL);
        Follow steps 3 to 9
STEP 3: WHILE (P2 (link) != NULL)
        Follow step 4 to 7
STEP 4:POLY (COEFF) = P1 (COEFF) *P2 (COEFF)
STEP 5: POLY (POWER) = P1 (POWER) + P2 (POWER)
STEP 6: NewNode = create Node (COEFF, POWER, POLY)
        Newnode=Poly (link)
        poly (link) = NULL
```

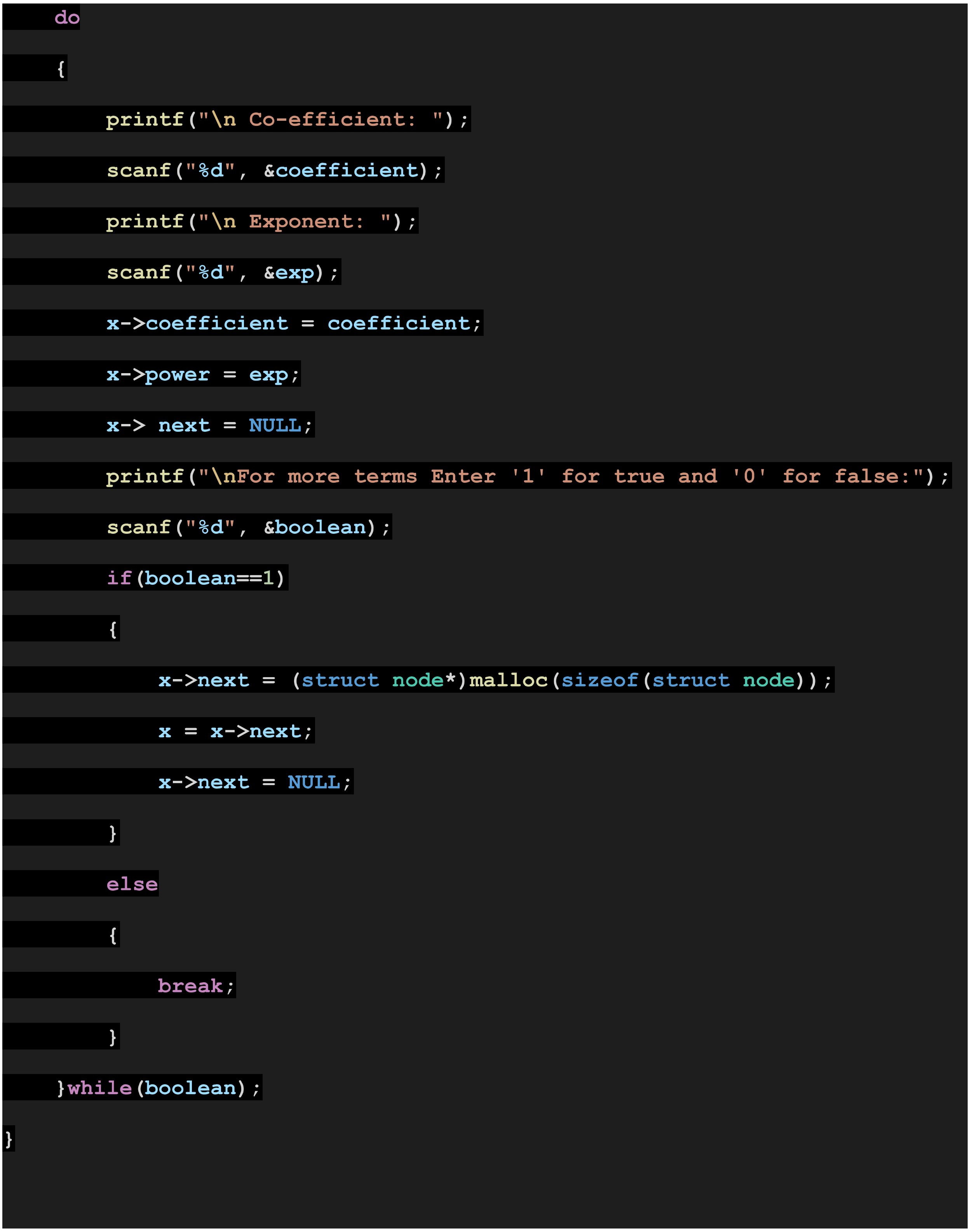
```
STEP 7: P2 = P2 (LINK)
STEP 8: RESTART P2.
STEP 9: P1 = P1(link).
STEP 10: END
```

TECHNOLOGY STACK USED:

VISUAL STUDIO CODE

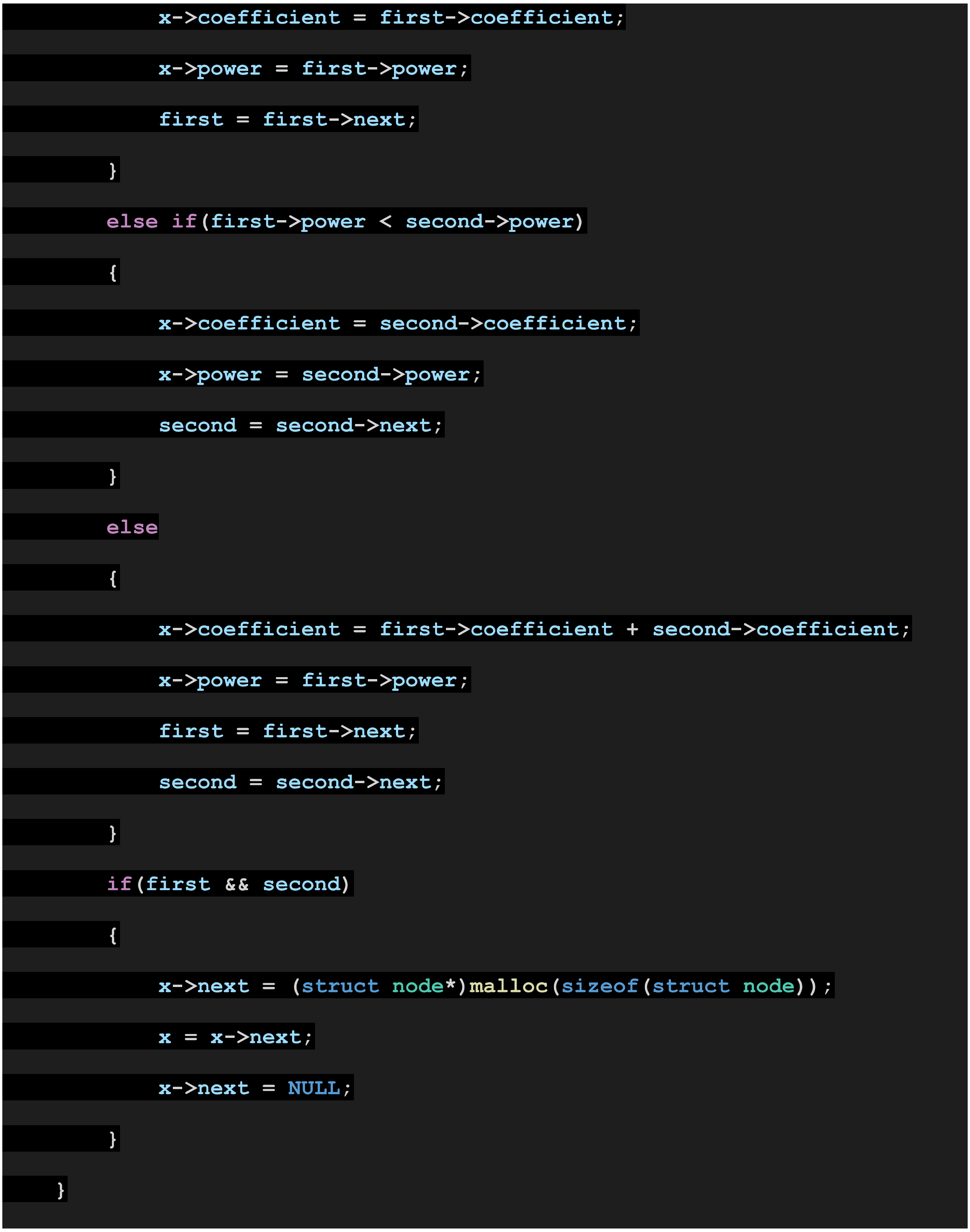
SOURCE CODE (OPTIONAL):

```
#include<stdio.h>

#include<stdlib.h>



struct node

{

    int coefficient;

    int power;

    struct node* next;

};



void read(struct node** polynomial)

{

    int coefficient, exp, boolean;

    struct node* x = (struct node*)malloc(sizeof(struct node));

    *polynomial = x;
```

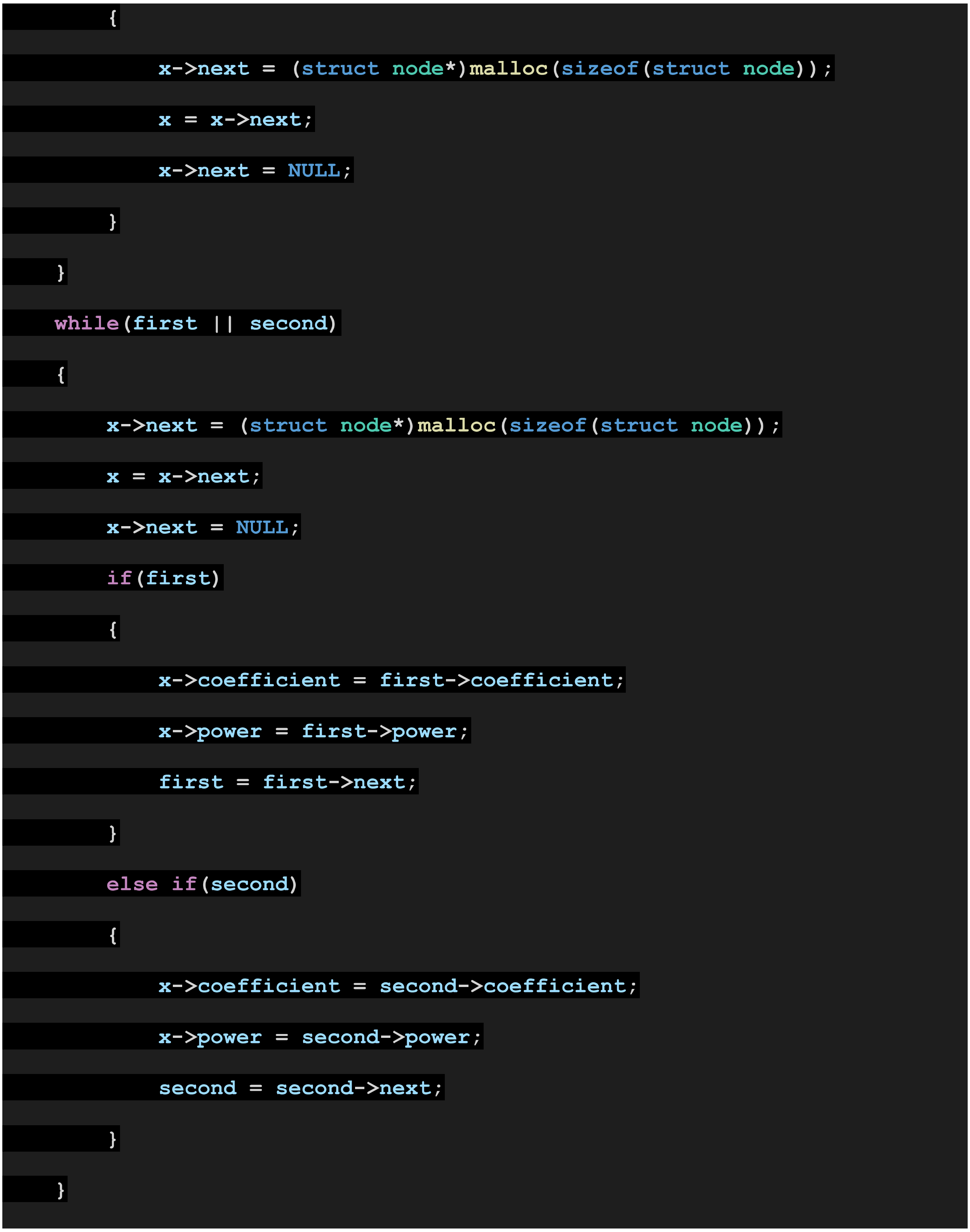


```
    do
    {
        printf("\n Co-efficient: ");
        scanf("%d", &coefficient);
        printf("\n Exponent: ");
        scanf("%d", &exp);
        x->coefficient = coefficient;
        x->power = exp;
        x-> next = NULL;
        printf("\nFor more terms Enter '1' for true and '0' for false:");
        scanf("%d", &boolean);
        if(boolean==1)
        {
            x->next = (struct node*)malloc(sizeof(struct node));
            x = x->next;
            x->next = NULL;
        }
        else
        {
            break;
        }
    }while(boolean);
}
```

```
void display(struct node* polynomial)
{
    printf("\nPolynomial expression is: ");
    while(polynomial != NULL)
    {
        printf("%dX^%d", polynomial->coefficient, polynomial->power);
        polynomial = polynomial->next;
        if(polynomial != NULL)
        printf("+");
    }
}




void addition(struct node** result, struct node* first, struct node*
second)
{
    struct node* x = (struct node*)malloc(sizeof(struct node));
    x->next = NULL;
    *result = x;
    while(first && second)
    {
        if(first->power > second->power)
        {
```



```
            x->coefficient = first->coefficient;
            x->power = first->power;
            first = first->next;
        }
        else if(first->power < second->power)
        {
            x->coefficient = second->coefficient;
            x->power = second->power;
            second = second->next;
        }
        else
        {
            x->coefficient = first->coefficient + second->coefficient;
            x->power = first->power;
            first = first->next;
            second = second->next;
        }
        if(first && second)
        {
            x->next = (struct node*)malloc(sizeof(struct node));
            x = x->next;
            x->next = NULL;
        }
    }
    while(first || second)
    {
        x->next = (struct node*)malloc(sizeof(struct node));
        x = x->next;
        x->next = NULL;
        if(first)
        {
            x->coefficient = first->coefficient;
            x->power = first->power;
            first = first->next;
        }
        else if(second)
        {
            x->coefficient = second->coefficient;
            x->power = second->power;
            second = second->next;
        }
    }
}



void subtract(struct node** result, struct node* first, struct node* second)
{
    struct node* x = (struct node*)malloc(sizeof(struct node));
    x->next = NULL;

    *result = x;

    while(first && second)

    {

        if(first->power > second->power)

        {

            x->coefficient = first->coefficient;

            x->power = first->power;

            first = first->next;

        }

        else if(first->power < second->power)

        {

            x->coefficient = second->coefficient;

            x->power = second->power;

            second = second->next;

        }

        else

        {

            x->coefficient = first->coefficient - second->coefficient;

            x->power = first->power;

            first = first->next;

            second = second->next;

        }

        if(first && second)
```

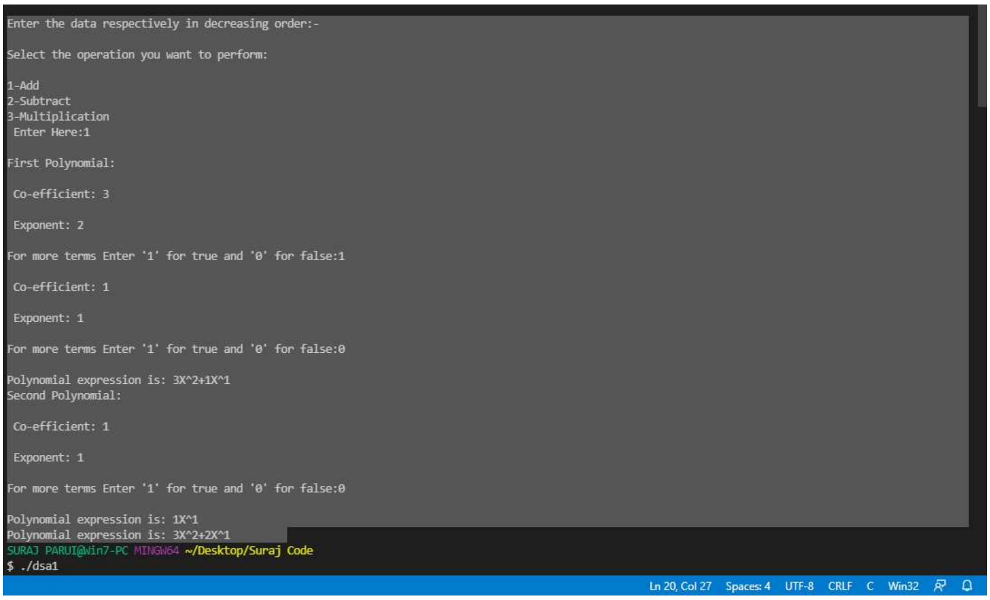


```
        {
            x->next = (struct node*)malloc(sizeof(struct node));
            x = x->next;
            x->next = NULL;
        }
    }
    while(first || second)
    {
        x->next = (struct node*)malloc(sizeof(struct node));
        x = x->next;
        x->next = NULL;
        if(first)
        {
            x->coefficient = first->coefficient;
            x->power = first->power;
            first = first->next;
        }
        else if(second)
        {
            x->coefficient = second->coefficient;
            x->power = second->power;
            second = second->next;
        }
    }
```

```
}



void multiply(struct node** result, struct node* first, struct node* second)
{
    struct node* x =(struct node*)malloc(sizeof(struct node));
    x->next = NULL;
    *result = x;
    if(first==NULL || second==NULL)
    {
        printf("0");
        return;
    }
    while(first!=NULL)
    {
        while(second!=NULL)
        {
            x->next = (struct node*)malloc(sizeof(struct node));
            x->next = NULL;
            x->coefficient = first->coefficient * second->coefficient;
            x->power = first->power + second->power;
            second = second->next;
            x = x->next;
        }
        first=first->next;
    }
}



int main()
{
    struct node* first = NULL;
    struct node* second = NULL;
    struct node* result = NULL;
    int sel;
    printf("\nEnter the data respectively in decreasing order:-\n");
    printf("\nSelect the operation you want to perform:\n");
    printf("\n1-Add\n2-Subtract\n3-Multiplication\n Enter Here:");
    scanf("%d",&sel);
    printf("\nFirst Polynomial:\n");
    read(&first);
    display(first);
    printf("\nSecond Polynomial:\n");
    read(&second);
    display(second);
    if(sel==1)
    {
        addition(&result, first, second);
        display(result);
    }
    else if(sel==2)
    {
        subtract(&result, first, second);
        display(result);
    }
    else if(sel==3)
        {
multiply(&result, first, second);
display(result);
}
else
{
printf("Wrong input");
}
return 0;
}
```

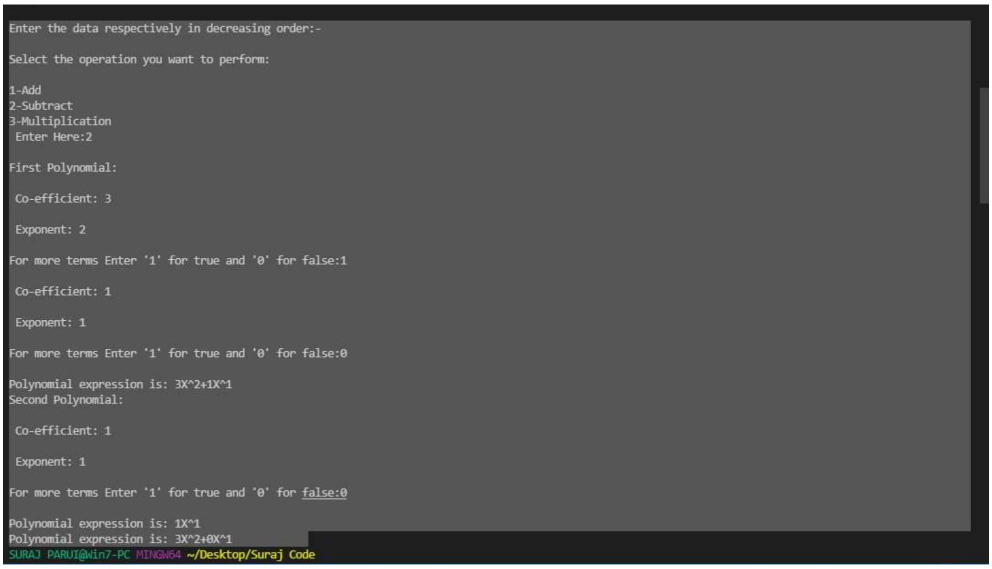
OBSERVATIONS / DISCUSSION OF RESULT:

ADDITION:



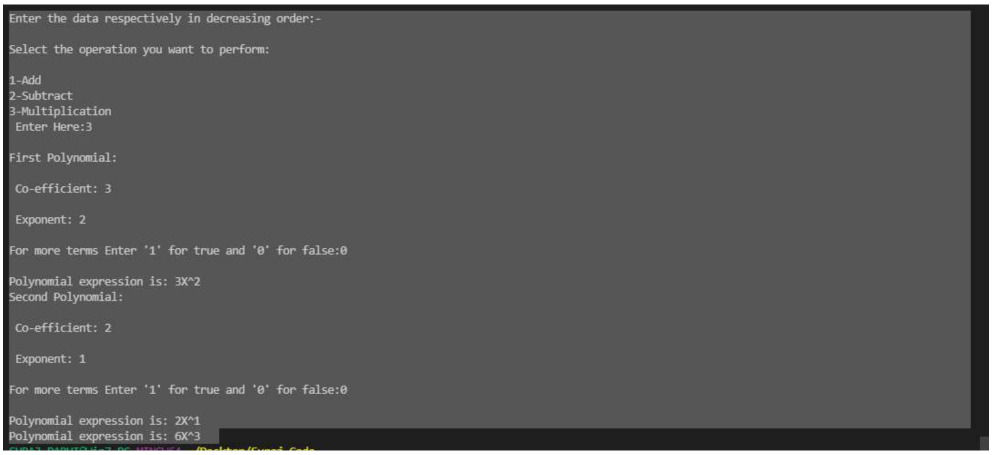
```
Enter the data respectively in decreasing order:-

Select the operation you want to perform:

1-Add
2-Subtract
3-Multiplication
 Enter Here:1

First Polynomial:

 Co-efficient: 3

 Exponent: 2

For more terms Enter '1' for true and '0' for false:1

 Co-efficient: 1

 Exponent: 1

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 3X^2+1X^1
Second Polynomial:

 Co-efficient: 1

 Exponent: 1

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 1X^1
Polynomial expression is: 3X^2+2X^1
SURAJ PARUI@Win7-PC MINGW64 ~/Desktop/Suraj Code
$ ./dsa1
```

SUBTRACTION:



```
Enter the data respectively in decreasing order:-

Select the operation you want to perform:

1-Add
2-Subtract
3-Multiplication
 Enter Here:2

First Polynomial:

 Co-efficient: 3

 Exponent: 2

For more terms Enter '1' for true and '0' for false:1

 Co-efficient: 1

 Exponent: 1

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 3X^2+1X^1
Second Polynomial:

 Co-efficient: 1

 Exponent: 1

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 1X^1
Polynomial expression is: 3X^2+0X^1
SURAJ PARUI@Win7-PC MINGW64 ~/Desktop/Suraj Code
```

MULTIPLICATION:



```
Enter the data respectively in decreasing order:-

Select the operation you want to perform:

1-Add
2-Subtract
3-Multiplication
 Enter Here:3

First Polynomial:

 Co-efficient: 3

 Exponent: 2

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 3X^2
Second Polynomial:

 Co-efficient: 2

 Exponent: 1

For more terms Enter '1' for true and '0' for false:0

Polynomial expression is: 2X^1
Polynomial expression is: 6X^3
```

CONCLUSION:THUS WE CAN PERFORM BASIC OPERATORS LIKE ADDITION, SUBTRACTION AND MULTIPLICATION USING LINKED LIST.

REFERENCES:

(List the references as per format given below and citations to be included the document)

R. F. Gilberg and B. A. Forouzan, Data Structures – A Pseudocode Approach with C, 2nd Edition, Cengage Learning, 2005