**Batch: A1 Roll No: 1911004**

**Experiment No 5**

**Grade: AA / AB / BB / BC / CC / CD /DD**

|  |
| --- |
| **Title:**  Implementation of polynomial operations using linked list |

**Objective:** To understand the application of Linked list to perform various mathematical operations like addition and subtraction on high power equations easily.

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
| CO2 | Apply linear and nonlinear data structure in application development |

**Books/ Journals/ Websites referred:**

1. Data Structures A Pseudocode Approach with C, Richard F. Gilberg & Behrouz A. Forouzan, second edition, CENGAGE learning.
2. Data Structures Using C & C++, Rajesh K. Shukla, Wiley- india.

**Abstract**:- Polynomials are used in both scientific and business-oriented problems. General-Purpose programming languages like C, C++ do not have built-in data types or functions to manipulate polynomials directly. Rather, it is common to represent polynomials using Arrays or Linked Lists. For any polynomial operation linked list representation is most suitable to deal with and handle various mathematical operations on it.

**Linked list representation of polynomial of single variables:**

Each polynomial can be represented as a list of coefficients and their powers with help of linked list representation. The first node in the list represent first term in the polynomial, the second node represents the second term, and so forth. Here the memory allocated for each node of the Linked List is divided in to 3 fields. The first field is the term’s coefficient. The second field is the term’s power, the third field is a pointer to the next term. While maintaining the polynomial it is assumed that exponent of each successive term is less than that of previous term. Or else before performing the operations we can sort the list as per the power and then do addition and subtraction because the user can entire a higher exponent term at the start by mistake.

**Related Theory:**

**Link-List :**

**It is a Data Structure which stores each data element as a data member in a struct node of the linked-list. In normal Link list we have a struct node which has 2 data member**

**struct node{**

**int nodeData;**

**struct node \*nextAddress; };**

**For this Polynomial expression we create**

**struct node {**

**int power, coefficient ;**

**struct node \*next; } ;**

**Polynomial Equation is a Linear Equation that consists of terms having variables with particular powers & coefficients.**

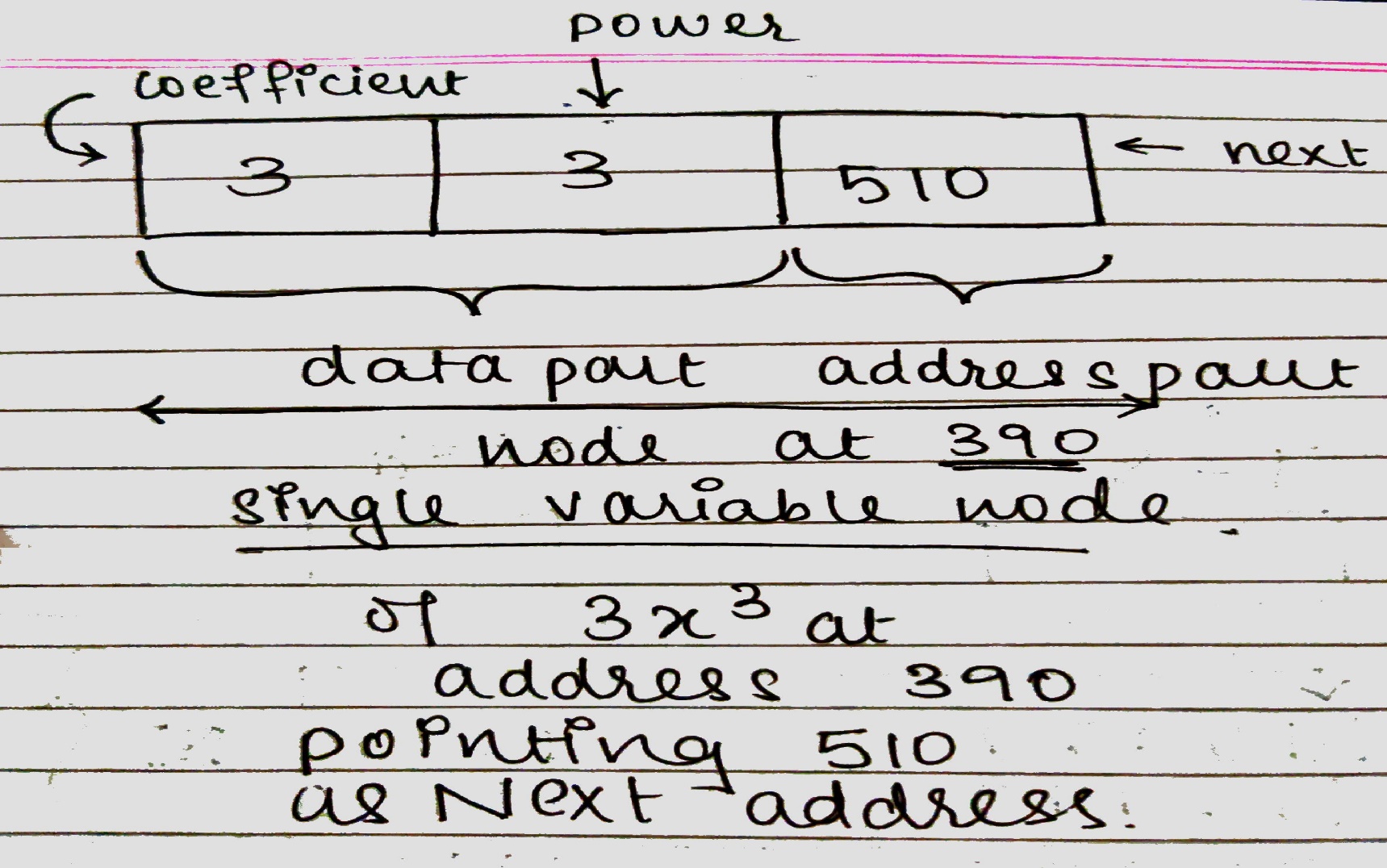
**1\* X3  + 3 \* X2 + 3\*X+ 1 \* X0**

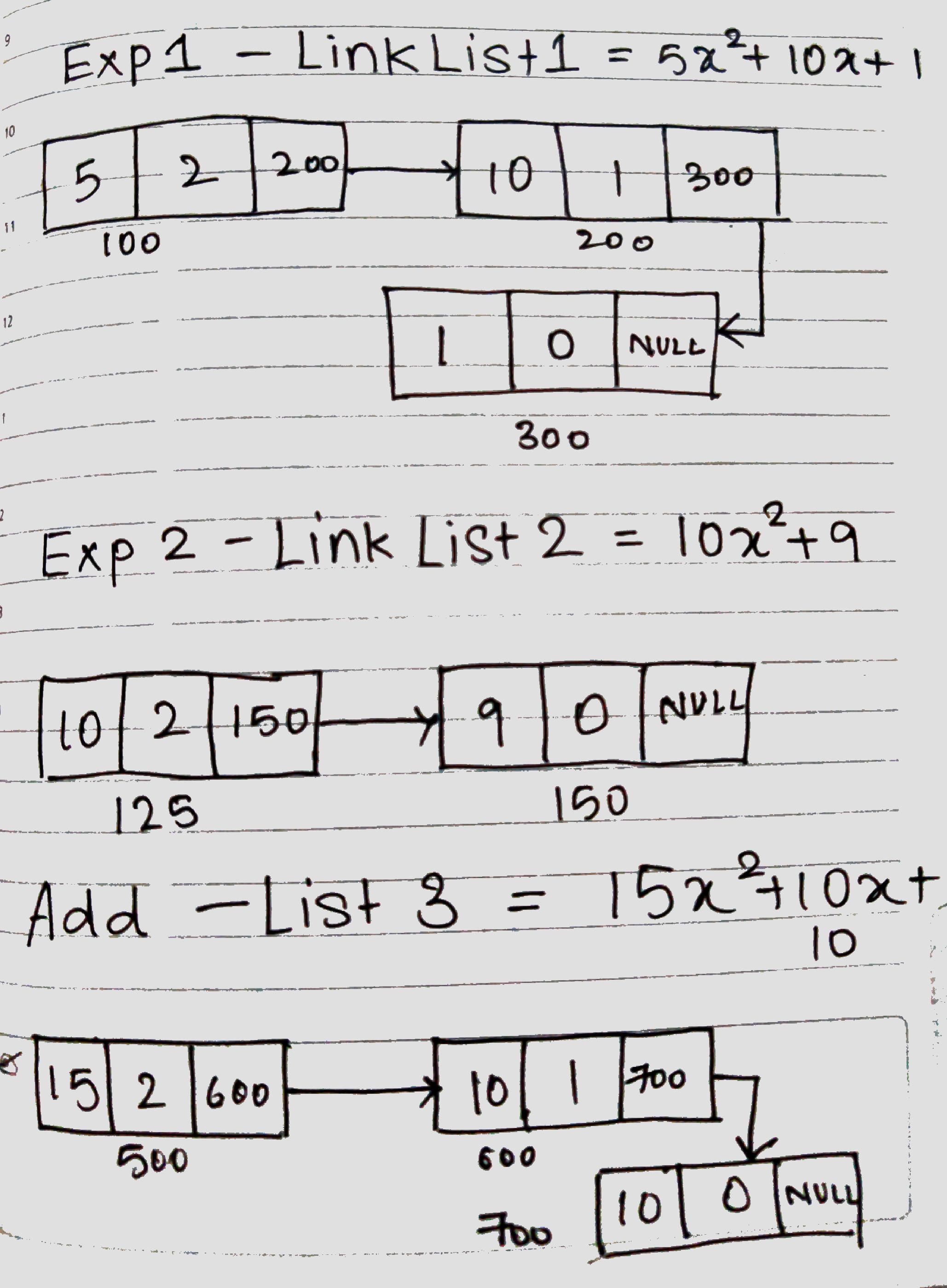
**In ADDTION / SUBTRACTION of two polynomials that are stored in form of a linked list.**

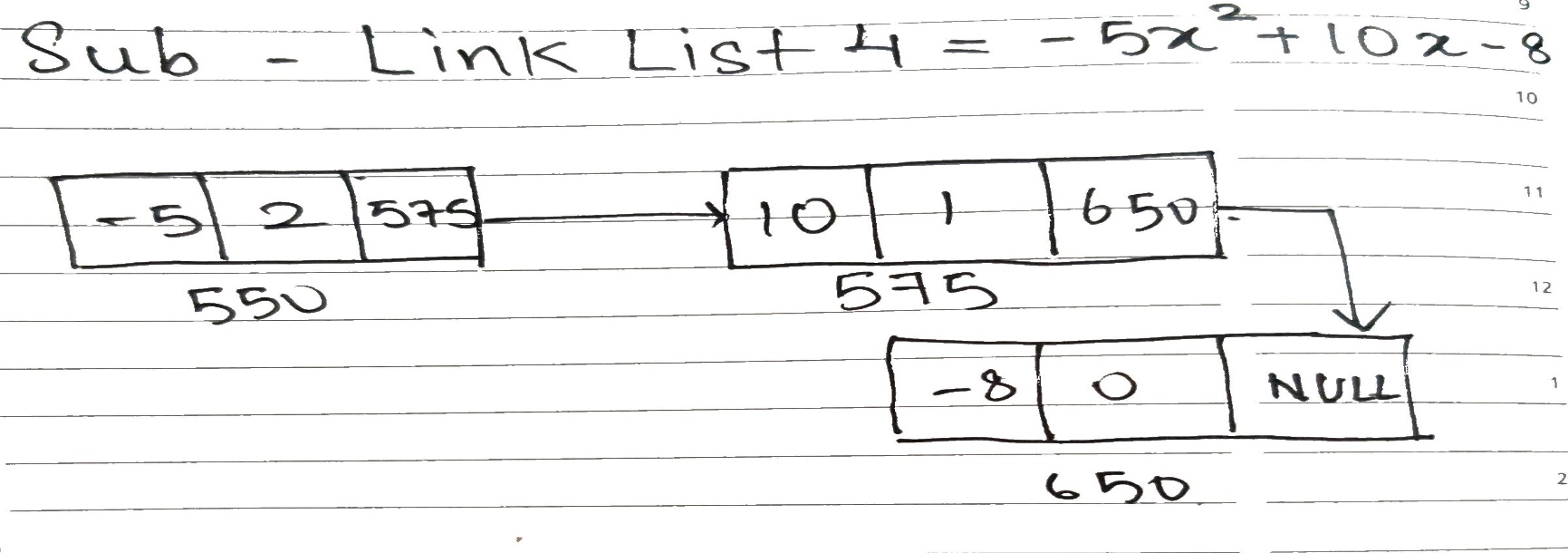
* **The powers of the terms(nodes) are checked and the we add / subtract the coefficients of term (node i.e. variables) with the same power.**
* **Else with unequal power we just copy the data in answer node of answer link-list**

**Thus we solve the Polynomial Equation Arithmetic**

**Diagram for linked representation of polynomial of single variables:**







**Algorithm for addition of two polynomials:**

1. **Start**
2. **Create the struct node with 2 data & 1 address part in node**
3. **1st data part will contain coefficient**
4. **2nd data part will have power**
5. **1 address next part for address of next node**
6. **We ask user to enter both polynomials 1 by 1 in pExp(struct node \*\* node)**
7. **Then we print those Polynomials using display()**
8. **in addPoly(struct node \*\* result, struct node \* P1, struct node \* P2) we add the polynomials**
9. **start from the first node if the value of power is greater we copy that node with higher power in result node and increment our node pointer**
10. **if both nodes have same power we add the coefficients of both the nodes in result node and increment node pointer**
11. **if the link list is traversed then we display the result else we repeat a & b**
12. **End**

**Source Code:**

**#include<stdio.h>**

**#include<stdlib.h>**

**struct node {**

**int coeff;**

**int pow;**

**struct node \* next;**

**};**

**void pExp(struct node \*\* node) { // pExp() takes the polynomial exp lists**

**int coeff, pow,flag;**

**struct node \* ptr; //To hold the temporary last address & pointer**

**ptr = (struct node \*) malloc(sizeof(struct node)); //creating first node**

**\*node = ptr;**

**do {**

**//enter the data**

**printf("\n Enter Coefficient : ");**

**scanf("%d", &coeff);**

**ptr->coeff = coeff;**

**printf("\n Enter Power : ");**

**scanf("%d", &pow);**

**ptr->pow = pow;**

**ptr->next = NULL;**

**//check if more nodes needed**

**printf("\nAre there more terms of the polynomial? (Yes = 1 && No = 0): ");**

**scanf("%d", &flag);**

**if(flag) {**

**ptr->next = (struct node \*) malloc(sizeof(struct node));**

**ptr = ptr->next;**

**ptr->next = NULL;**

**}**

**} while (flag);**

**}**

**void display(struct node \* node) { // display() displays the exp1,exp2,addAns,subAns lists**

**printf("\nThe polynomial expression is:\t\t");**

**while(node != NULL) {**

**printf("%dX^%d", node->coeff, node->pow);**

**node = node->next;**

**if(node != NULL)**

**printf(" + ");**

**}**

**}**

**void addPoly(struct node \*\* result, struct node \* P1, struct node \* P2) {**

**// addPoly() to add up the coefficient of the same powers of the polynomial expression struct node \* ptr; //Temporary storage for the linked list & pointer**

**ptr = (struct node \*) malloc(sizeof(struct node));**

**ptr->next = NULL;**

**\*result = ptr;**

**//Loop when both lists have values**

**while(P1 && P2) {**

**if (P1->pow > P2->pow) {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff;**

**P1 = P1->next;**

**}**

**else if (P1->pow < P2->pow) {**

**ptr->pow = P2->pow;**

**ptr->coeff = P2->coeff;**

**P2 = P2->next;**

**}**

**else {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff + P2->coeff;**

**P1 = P1->next;**

**P2 = P2->next;**

**}**

**if(P1 && P2) {**

**ptr->next = (struct node \*) malloc(sizeof(struct node));**

**ptr = ptr->next;**

**ptr->next = NULL;**

**}**

**}**

**//Loop when any 1 has value**

**while(P1 || P2) {**

**ptr->next = (struct node \*) malloc(sizeof(struct node));**

**ptr = ptr->next;**

**ptr->next = NULL;**

**if(P1) {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff;**

**P1 = P1->next;**

**}**

**if(P2) {**

**ptr->pow = P2->pow;**

**ptr->coeff = P2->coeff;**

**P2 = P2->next;**

**}**

**}**

**printf("\n Added! ");**

**}**

**void subPoly(struct node \*\* result, struct node \* P1, struct node \* P2) {**

**// subPoly() to subtract the coefficient of the same powers of the polynomial expression**

**struct node \* ptr; //Temporary storage for the linked list & also pointer**

**ptr = (struct node \*) malloc(sizeof(struct node));**

**ptr->next = NULL;**

**\*result = ptr;**

**//Loop when both lists have values**

**while(P1 && P2) {**

**if (P1->pow > P2->pow) {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff;**

**P1 = P1->next;**

**}else if (P1->pow < P2->pow) {**

**ptr->pow = P2->pow;**

**ptr->coeff = P2->coeff;**

**P2 = P2->next;**

**}else {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff - P2->coeff;**

**P1 = P1->next;**

**P2 = P2->next;**

**}**

**if(P1 && P2) {**

**ptr->next = (struct node \*) malloc(sizeof(struct node));**

**ptr = ptr->next;**

**ptr->next = NULL;**

**}**

**}**

**//Loop when any 1 has values**

**while(P1 || P2) {**

**ptr->next = (struct node \*) malloc(sizeof(struct node));**

**ptr = ptr->next;**

**ptr->next = NULL;**

**if(P1) {**

**ptr->pow = P1->pow;**

**ptr->coeff = P1->coeff;**

**P1 = P1->next;**

**}if(P2) {**

**ptr->pow = P2->pow;**

**ptr->coeff = P2->coeff;**

**P2 = P2->next;**

**}**

**}**

**printf("\n Subtracted! ");**

**}**

**int main() {**

**int ch;**

**do {**

**struct node \* P1, \* P2, \* P3,\* P4;**

**printf("\nEnter 1st expression\n");**

**pExp(&P1);**

**printf("\nStored the 1st expression");**

**display(P1);**

**printf("\nEnter 2nd expression\n");**

**pExp(&P2);**

**printf("\nStored the 2nd expression");**

**display(P2);**

**addPoly(&P3, P1, P2);**

**display(P3);**

**subPoly(&P4, P1, P2);**

**display(P4);**

**printf("\n Do You Want ADD & SUB AGAIN? \t1:Yes || 0:No \n CHOICE : ");**

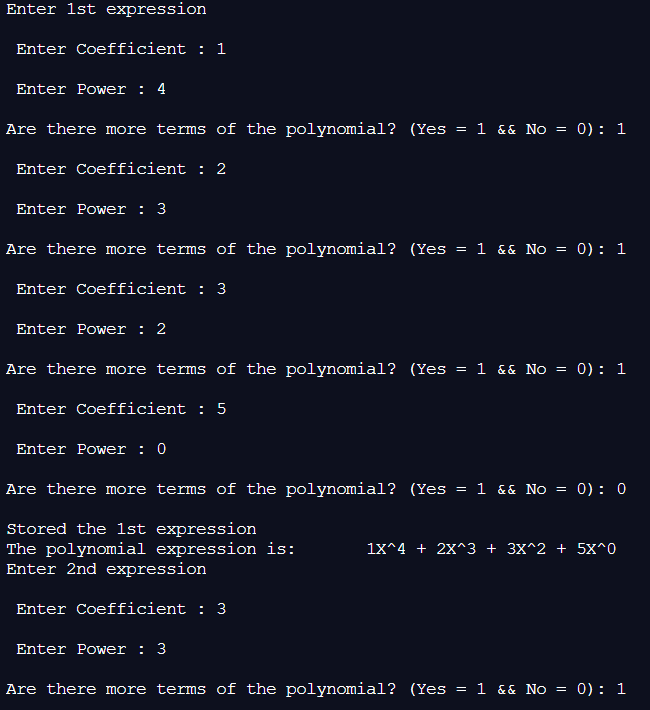
**scanf("%d", &ch);**

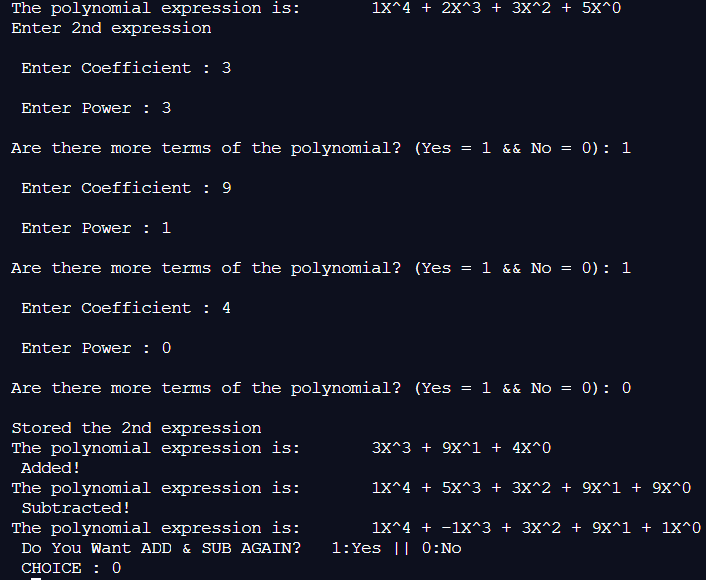
**} while (ch!= 0);**

**return 0;**

**} //main() ends**

**Output screenshots:**





**Implementation Details:**

1. **Enlist all the Steps followed and various options explored**
2. **Start**
3. **Create the struct node with 2 data & 1 address part in node because**

* **1st data part store the power**
* **2nd data stores coefficient of term**
* **address part stores address of next node**

1. **We ask user to enter both polynomials 1 by 1 in pExp(struct node \*\* node)**
2. **Then we print those Polynomials using display()**
3. **in addPoly(struct node \*\* result, struct node \* P1, struct node \* P2) we add the polynomials**

* **start from the first node if the value of power is greater we copy that node with higher power in result node and increment our node pointer**
* **if both nodes have same power we add the coefficients of both the nodes & store result node and increment node pointer**
* **if the link list is traversed then we display the result else we repeat a & b using display()**

1. **in subPoly(struct node \*\* result, struct node \* P1, struct node \* P2) we subtract the polynomials**

* **start from the first node if the value of power is greater we copy that node with higher power in result node and increment our node pointer**
* **if both nodes have same power we subtract 1 coefficient from other of the nodes & store result node and increment node pointer**
* **if the link list is traversed then we display the result else we repeat a & b using display()**

1. **End**

**2. Explain your program logic, classes and methods used.**

**void pExp(struct node \*\* node)**

* **pExp() takes the polynomial exp lists**
* **we ask for the powers & coefficients of the single variable nodes of each term of the polynomial**
* **till user donesnt say that expression is over we take inputs**
* **this is done twice for 2 expression**

**void display(struct node \* node)**

* **display() displays the exp1,exp2,addAns,subAns lists**
* **till the node pointer is not null/0 we display all the node’s data part**
* **this is done for all the lists**

**void addPoly(struct node \*\* result, struct node \* P1, struct node \* P2)**

* **addPoly() to add up the coefficient of the same powers of the polynomial expression**
* **we compare the power and the we add up our coefficients of respective powers**
* **in case for particular expression a power is missing the that terms coefficient is taken 0 and added**

**void subPoly(struct node \*\* result, struct node \* P1, struct node \* P2)**

* **subPoly() to subtract the coefficient of the same powers of the polynomial expression**
* **we compare the power and the we subtract up our coefficients of respective powers**
* **in case for particular expression a power is missing the that terms coefficient is taken 0 and subtracted**

**int main()**

* **Calls above functions to perform add, sub of polynomials**

1. **Explain the Importance of the approach followed by you**

* In the approach followed

1. We take the input from user for both the Polynomial Equation
2. We then display both
3. We iterate through both the link list in order to check power of each of the node in each link list
4. if powers are same we add/ subtract the coefficients
5. if power are same we just copy them as it is
6. Thus following this approach we traverse through both of our Polynomial Equation to get our Add / Subtract Answers.
7. If n = complexity of the Polynomial Equation 1 & m= complexity of the Polynomial Equation 2

then total complexity for solving Add/ subtract is O (n + m)

1. Though 2\*int + 1\* struct node type bytes is required this is the easiest approach to find Arithmetic on Polynomials .

**Conclusion:-**

**We understood the concept of Link-List along with Concept of Adding & Subtracting of the Polynomials using Link-List ; thus implemented it in C , Obtained the needed correct output.**