PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester :** I

Name of the student: Samruddhi Ramswami Bansode. **PRN :** 124B1B019

Department: Computer Engineering **Division :** A

Course Name : Data Structure Lab

Course Code: BCE23PC02

Completion Date :

Assignment No. 5

Problem Statement:

Write a program to perform Polynomial Addition using Linked Lists

- Each term is a node (with coefficient and power).
- Add two polynomials represented by linked lists.

Source Code :

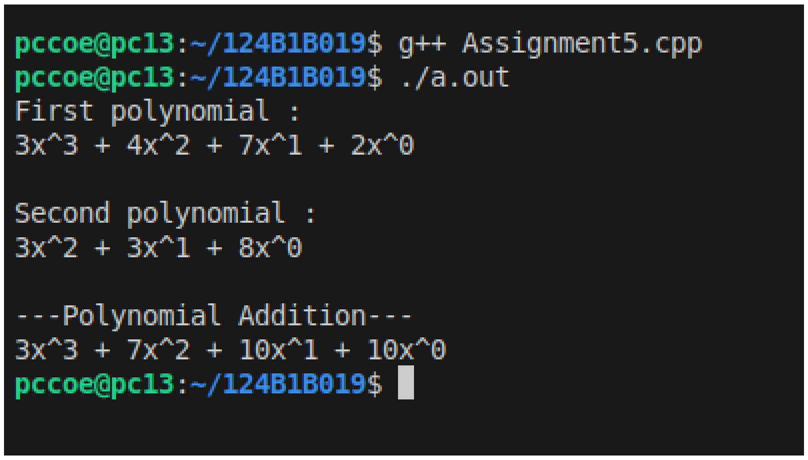
```
#include<iostream>
using namespace std;
class Node{
   public:
   Node* next;
   int coe, pow;
   Node(){
      next = NULL;
      coe = 0;
      pow = 0;
   }
```

```
    Node(int c, int p){
        coe = c;
        pow = p;
        next = NULL;
    }
};
class LL{
    public:
    Node* head;
    LL(){   head = NULL;  }
    void Insert_end(int c, int p);
    void Add_polynomial(LL l1, LL l2);
    void print();
};

// to insert new node in LL
void LL :: Insert_end(int c, int p){
    Node* nn = new Node(c,p);
    if(head == NULL){
        head = nn;
    } else {
        Node* t = head;
        while(t-> next != NULL){
            t = t-> next;
        }
        t->next = nn;
    }
}

//Addition of two polynomial represented by LL
```

```
void LL :: Add_polynomial(LL l1, LL l2){
   Node* p1 = l1.head;
   Node* p2 = l2.head;
   int c, p;

   while(p1 != NULL && p2 != NULL){
      if(p1-> pow  >  p2-> pow){
         c = p1-> coe;
         p = p1-> pow;
         p1 = p1-> next;
      }
      else if(p1-> pow  <  p2-> pow){
         c = p2-> coe;
         p = p2-> pow;
         p2 = p2-> next;
      }
      else {
         c = p1-> coe + p2-> coe;
         p = p1-> pow;
         p1 = p1-> next;
         p2 = p2-> next;
      }
      Insert_end(c,p);
   }
   while(p1 != NULL){
      c = p1-> coe;
      p = p1 -> pow;
      p1 = p1-> next;
      Insert_end(c,p);
   }
```

```
    while(p2 != NULL){
        c = p2-> coe;
        p = p2 -> pow;
        p1 = p2-> next;
        Insert_end(c,p);
    }
}

// Print the polynomial
void LL::print() {
    Node* t = head;
    Node* p = head;

    if (head == NULL) {
        cout << "LL is Empty" << endl;
    } else {
        // Print coefficients
        while (t->next != NULL && p-> next != NULL) {
            cout << t->coe << "x^" << p-> pow << " + ";
            t = t->next;
            p = p->next;
        }
        cout << t->coe << "x^" << p-> pow << " ";
    }
}

int main(){
    LL l1, l2, res;

    l1.Insert_end(3, 3);
    l1.Insert_end(4, 2);
    l1.Insert_end(7, 1);
    l1.Insert_end(2, 0);

    l2.Insert_end(3, 2);
    l2.Insert_end(3, 1);
    l2.Insert_end(8, 0);

    res.Add_polynomial(l1, l2);
    cout << "First polynomial : \n";
    l1.print();
    cout << "\n\nSecond polynomial : \n";
    l2.print();

    cout << "\n\n---Polynomial Addition---\n";
    res.print();
    cout << endl;
    return 0;
}
```

Screen Shot of Output :



```
pccoe@pc13:~/124B1B019$ g++ Assignment5.cpp
pccoe@pc13:~/124B1B019$ ./a.out
First polynomial :
3x^3 + 4x^2 + 7x^1 + 2x^0

Second polynomial :
3x^2 + 3x^1 + 8x^0

---Polynomial Addition---
3x^3 + 7x^2 + 10x^1 + 10x^0
pccoe@pc13:~/124B1B019$
```

Conclusion:

The assignment demonstrates the use of linked lists to represent and add polynomials. Each node stores a coefficient and power, allowing terms to be managed dynamically. Traversal and comparison of nodes enable addition of like terms while preserving polynomial order. This approach highlights the efficiency of linked lists in handling variable-sized data and performing operations without the limitations of static arrays.