#include <stdio.h>

#include <stdlib.h>

#include <math.h>

struct Node

{

    int power;

    float coe;

    struct Node \*next;

};

typedef struct Node \*node;

node insert\_polynomial(node head, int power, float coe)

{

    node newNode, temp;

    newNode = (node)malloc(sizeof(struct Node));

    newNode->power = power;

    newNode->coe = coe;

    newNode->next = NULL;

    if (head == NULL)

    {

        head = newNode;

        head->next = newNode;

        return (head);

    }

    if (power < head->power)

    {

        newNode->next = head->next;

        head->next = newNode;

        head = newNode;

        return (head);

    }

    if (power == head->power)

    {

        head->coe += coe;

        return (head);

    }

    temp = head;

    while (temp->next != head && power <= temp->next->power)

    {

        temp = temp->next;

    }

    if (newNode->power == temp->power)

    {

        temp->coe += coe;

    }

    else

    {

        newNode->next = temp->next;

        temp->next = newNode;

    }

    return (head);

}

void display(node head)

{

    node temp;

    temp = head->next;

    printf("\n");

    do

    {

        printf("%6.2f x^%d ", temp->coe, temp->power);

        temp = temp->next;

        if (temp > head->next)

        {

            printf("+ ");

        }

    } while (temp != head->next);

}

node accept(node head)

{

    int n, power;

    float coe;

    printf("\n Enter the NUMBER OF TERMS : ");

    scanf("%d", &n);

    printf("\n Enter the terms as COEFFICENT & POWER :");

    for (int i = 0; i < n; i++)

    {

        scanf("%f%d", &coe, &power);

        head =

            insert\_polynomial(head, power, coe);

    }

    return (head);

}

node add(node p1, node p2)

{

    node p, head = NULL;

    int power;

    float

        coeffiecent;

    p = p1->next;

    do

    {

        head = insert\_polynomial(head, p->power, p->coe);

        p = p->next;

    } while (p != p1->next);

    p = p2->next;

    do

    {

        head = insert\_polynomial(head, p->power, p->coe);

        p = p->next;

    } while (p != p2->next);

    return (head);

}

node multi(node p1, node p2)

{

    node head = NULL, head1, head2 = p2->next;

    do

    {

        head1 = p1->next;

        do

        {

            head = insert\_polynomial(head, head1->power + head2->power, head1->coe \* head2->coe);

            head1 = head1->next;

        } while (head1 != p1->next);

        head2 = head2->next;

    } while (head2 != p2->next);

    return (head);

}

float eval(node head, float x)

{

    float value = 0;

    node p;

    p = head->next;

    do

    {

        value += p->coe \* pow(x, p->power);

        p = p->next;

    } while (p != head->next);

    return (value);

}

int main()

{

    node p1 = NULL, p2 = NULL, p3 = NULL;

    int choice;

    float value, x;

    do

    {

        printf("\n 1.Create polynomials \n 2.Display polynomials\n 3.Add polynomials \n 4.Multiply polynomials\n 5.Evaluate polynomial\n 6.Exit");

        printf("\n Enter your choice :");

        scanf("%d", &choice);

        switch (choice)

        {

        case 1:

            printf("\n Enter the first polynomial :");

            p1 = accept(p1);

            printf("\n Enter the second polynomial :");

            p2 = accept(p2);

            break;

        case 2:

            printf("\n First polynomial :");

            display(p1);

            printf("\n Second polynomial:");

            display(p2);

            break;

        case 3:

            p3 = add(p1, p2);

            printf("\nAddition of polynomials =");

            display(p3);

            break;

        case 4:

            p3 = multi(p1, p2);

            printf("\nMultiplication of polynomial =");

            display(p3);

            break;

        case 5:

            printf("\nEnter the value of x:");

            scanf("%f", &x);

            value = eval(p1, x);

            printf("\nValue :%f", value);

            break;

        }

    } while (choice != 6);

    return 0;

}