DS LAB 6: Add 2 polynomials using Linked List

Name : Vivian Vijay Ludrick

Branch : SE Comps A Batch C

Roll No: 9914

Code:

#include <stdio.h>

#include <stdlib.h>

*// Monomial Node creation*

typedef struct Node

{

int coeff; *// to store coeff*

int expo; *// to store expo*

struct Node \*next; *// next node*

} Node;

*// linkedList of Polynomial*

typedef struct

{

Node \*start;

} LL;

*// create a polynomial*

*// takes the input in the create function itself*

void create(LL \**l*)

{

Node \*p, \*prev;

int ch;

do

{

*// create monomial and append to polynomial*

p = (Node \*)malloc(sizeof(Node));

printf("Enter the coeff:\n");

scanf("%d", &p->coeff);

printf("Enter its expo:\n");

scanf("%d", &p->expo);

p->next = NULL;

if (*l*->start == NULL) *// first node*

{

*l*->start = p; *// point it by start*

}

else

{

prev->next = p;

}

prev = p; *// current node in this iteration becomes previous node in next iteration*

*// continue the creation of the polynomial*

printf("Enter 1 to continue creation of list and 0 to end it:\n");

scanf("%d", &ch);

} while (ch == 1);

}

*// display the polynomial*

void display(LL *l*)

{

Node \*ptr;

ptr = *l*.start;

while (ptr != NULL)

{

printf(" %dx^%d", ptr->coeff, ptr->expo);

if (ptr->next != NULL)

printf(" + "); *// separation using '+' sign; edge case: the sign of the coefficient is negative(can solve this by putting if else)*

ptr = ptr->next;

}

printf("\n");

}

*// add a monomial to the existing polynomial*

void append(LL \**ptr*, int *coeff*, int *expo*)

{

Node \*q, \*p;

p = (Node \*)malloc(sizeof(Node));

p->coeff = *coeff*;

p->expo = *expo*;

p->next = NULL;

if (*ptr*->start == NULL)

{

*ptr*->start = p;

}

else

{

q = *ptr*->start;

while (q->next != NULL)

{

q = q->next;

}

q->next = p; *// append p to last*

}

}

*// add two polynomials and store in the first one*

void add(LL \**result*, LL *poly1*, LL *poly2*)

{

int sum;

Node \*ptr1 = *poly1*.start, \*ptr2 = *poly2*.start;

*// ptr1 for traversing polynomial 1*

*// ptr2 for traversing polynomial 2*

while (ptr1 != NULL && ptr2 != NULL) *// if two nodes in both polynomials are present*

{

if (ptr1->expo == ptr2->expo) *// if expos of terms are the same*

{

sum = ptr1->coeff + ptr2->coeff; *// add coefficients*

append(*result*, sum, ptr1->expo); *// add a node in the result polynomial*

ptr1 = ptr1->next;

ptr2 = ptr2->next;

}

else if (ptr1->expo > ptr2->expo) *// if the expo is greater, then append a node in the result polynomial with the highest expo*

{

append(*result*, ptr1->coeff, ptr1->expo);

ptr1 = ptr1->next;

}

else *// if (ptr1->expo < ptr2->expo)*

{

append(*result*, ptr2->coeff, ptr2->expo);

ptr2 = ptr2->next;

}

}

if (ptr1 == NULL) *// if the first polynomial is exhausted*

{

while (ptr2 != NULL) *// copy lower-order terms of the second polynomial to the result*

{

append(*result*, ptr2->coeff, ptr2->expo);

ptr2 = ptr2->next;

}

}

if (ptr2 == NULL) *// if the second polynomial is exhausted*

{

while (ptr1 != NULL) *// copy the first polynomial to the result*

{

append(*result*, ptr1->coeff, ptr1->expo);

ptr1 = ptr1->next;

}

}

}

int main()

{

int option;

LL poly1, poly2, result; *// three polynomials*

poly1.start = NULL;

poly2.start = NULL;

result.start = NULL;

*// Menu*

do

{

printf("Enter the option:1.Create polynomial 1 2.Create polynomial 2 3.Add polynomial 1 and 2 4.Exit\n");

scanf("%d", &option);

switch (option)

{

case 1:

create(&*poly1*);

printf("First polynomial created:\n");

display(poly1);

break;

case 2:

create(&*poly2*);

printf("Second polynomial created:\n");

display(poly2);

break;

case 3:

add(&*result*, poly1, poly2);

printf("Resultant polynomial:\n");

display(result);

break;

case 4:

break;

default:

printf("Enter a valid option.\n");

}

} while (option != 4);

return 0;

}

OUTPUT:



