//ADDING TWO POLYNOMIAL USING LINKED LIST

#include <stdio.h>

#include <stdlib.h>

struct Node {

int coefficient;

int exponent;

struct Node \*link;

};

struct Node \*read();

struct Node \*add(struct Node \*, struct Node \*);

void display(struct Node \*);

struct Node \*insertEnd(struct Node \*, struct Node \*);

int main() {

struct Node \*first, \*second;

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

first = read();

printf("\nThe first polynomial is: ");

display(first);

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

second = read();

printf("\nThe second polynomial is: ");

display(second);

printf("\nThe sum of two polynomials are: ");

display(add(first, second));

return 0;

}

struct Node \*insertEnd(struct Node \*HEAD, struct Node \*newNode) {

if (HEAD == NULL)

return newNode;

struct Node \*ptr = HEAD;

while (ptr->link != NULL)

ptr = ptr->link;

ptr->link = newNode;

newNode->link = NULL;

return HEAD;

}

void display(struct Node \*polynomial) {

struct Node \*ptr;

ptr = polynomial;

while (ptr != NULL) {

printf("%dX^%d ", ptr->coefficient, ptr->exponent);

ptr = ptr->link;

if (ptr != NULL)

printf("+ "); // printf("%c ", ptr->coefficient > 0 ? '+' : '-');

}

printf("\n");

}

struct Node \*read() {

struct Node \*HEAD = NULL;

while (1) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node \*));

printf("Enter the coefficient: ");

scanf("%d", &newNode->coefficient);

printf("Enter the exponent: ");

scanf("%d", &newNode->exponent);

HEAD = insertEnd(HEAD, newNode);

int choice;

printf("Enter 0 to exit and 1 to continue adding more terms: ");

scanf("%d", &choice);

if (choice == 0)

return HEAD;

}

}

struct Node \*add(struct Node \*first, struct Node \*second) {

struct Node \*HEAD = NULL;

while (first != NULL && second != NULL) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node \*));

if (first->exponent > second->exponent) {

newNode->exponent = first->exponent;

newNode->coefficient = first->coefficient;

first = first->link;

} else if (second->exponent > first->exponent) {

newNode->exponent = second->exponent;

newNode->coefficient = second->coefficient;

second = second->link;

} else {

newNode->exponent = first->exponent;

newNode->coefficient = first->coefficient + second->coefficient;

first = first->link;

second = second->link;

}

HEAD = insertEnd(HEAD, newNode);

}

while (first != NULL || second != NULL) {

struct Node \*newNode = (struct Node \*)malloc(sizeof(struct Node \*));

if (first != NULL) {

newNode->exponent = first->exponent;

newNode->coefficient = first->coefficient;

first = first->link;

} else {

newNode->exponent = second->exponent;

newNode->coefficient = second->coefficient;

second = second->link;

}

HEAD = insertEnd(HEAD, newNode);

}

return HEAD;

}