Polynomial Addition and Evaluation

locked

Problem Submissions Leaderboard Discussions

Write a C program that takes 2 polynomials, adds them and evaluate it with an input value(x). If the resultant linked list is empty print -1.

You are required to fill in the code for the following functions:

-The 'input' function that inputs the terms of the polynomials.

-The 'sum' function that finds the sum of the polynomials and returns the resulting polynomial.

-The 'eval' function that evaluates the polynomial with the value x.

-The 'destroy function' that frees all the nodes in the linked list.

Note: Each node of a linked list for a polynomial stores each term of the polynomial(coefficient, degree). The terms in each polynomial are stored in decreasing order of their powers.

Input Format

Number of terms in polynomial 1(m) and polynomial 2(n) separated by a space

x (value to evaluate the resulting polynomial with)

Coefficient 1 Degree 1

.

.

.

m times

Coefficient 2 Degree 2

.

.

.

n times

Constraints

Constraints:

degree of the polynomial>=0

number of elements>=0

Output Format

sum of the resulting polynomial (if the resulting polynomial is not empty) else -1

Sample Input 0

```
2 2
2
2 1
3 0
3 1
4 0
```

Sample Output 0

```
17
```

Explanation 0

On adding the two polynomials and evaluating it with x=2, 17 is the answer obtained

Submissions: 62
Max Score: 10
Difficulty: Medium

Rate This Challenge:

☆☆☆☆☆

More

C

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>

typedef struct node
{
    int coeff;
    int pow;
    struct node* next;
}node;

void input(node** head,node** tail,int n);
node* sum(node* head1,node* head2);
int eval(node* head,int x);
void destroy(node **head);
node *getNode(int coeff, int pow);
void InsertFront(node **ph, int coeff, int pow);

int main()
{
    int m=0;
    int n=0;
    node* head1=NULL;
    node* head2=NULL;
    node* tail1=NULL;
    node* tail2=NULL;
    int x=0;//value of the variable for evaluation
    scanf("%d %d",&m,&n);
    scanf("%d",&x);
    input(&head1,&tail1,m);
    input(&head2,&tail2,n);
    node *headres=sum(head1,head2);
    int res=eval(headres,x);
    printf("%d",res);
      destroy(&head1);
     destroy(&head2);
    destroy(&headres);
    return 0;
```

```c
}

node *getNode(int coeff, int pow){
    node *temp=(node*)malloc(sizeof(node));
    temp->coeff=coeff;
    temp->pow=pow;
    temp->next=NULL;
    return temp;
}

void InsertFront(node **ph, int coeff, int pow){
    node *temp = getNode(coeff, pow);
    temp->next=*ph;
    *ph=temp;
}

void input(node** head,node** tail,int n)
{
    int coeff, pow;
    for(int i=0;i<n;i++){
        scanf("%d %d",&coeff, &pow);
        InsertFront(head, coeff, pow);
    }

}

node* sum(node* head1,node* head2)
{
    node* head3=NULL;
    while (head1 && head2)
    {
        if(head1->pow==head2->pow){
        InsertFront(&head3, head1->coeff+head2->coeff, head1->pow);
        head1=head1->next;
        head2=head2->next;
    }
    else if(head1->pow>head2->pow){
        InsertFront(&head3, head1->coeff,head1->pow);
        head1=head1->next;
    }
    else if(head1->pow<head2->pow){
        InsertFront(&head3, head2->coeff,head2->pow);
        head2=head2->next;
    }
    }

    while (head1)
    {
        InsertFront(&head3, head1->coeff,head1->pow);
        head1=head1->next;
    }

     while (head2)
    {
        InsertFront(&head3, head2->coeff,head2->pow);
        head2=head2->next;
    }

    return head3;
}


int eval(node* head,int x)
{
    int res=0;
    node *p=head;
    if(head==NULL){
        return -1;
    }
    while (p)
    {
        res+=(p->coeff)*pow(x,p->pow);
        p=p->next;
    }
    return res;

}
void destroy(node **head)
{
    while (*head) {
    node *temp = *head;
    *head = (*head)->next;
    free(temp);
  }

}
```

Line: 125 Col: 2

Upload Code as File Test against custom input Run Code Submit Code

Interview Prep | Blog | Scoring | Environment | FAQ | About Us | Support | Careers | Terms Of Service | Privacy Policy |