Rajalakshmi Engineering College

Name: SHIVANISREE K B
Email: 240701501@rajalakshmi.edu.in
Roll no: 240701501
Phone: 7358464804
Branch: REC
Department: I CSE FE
Batch: 2028
Degree: B.E - CSE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 23

Section 1 : Coding

1. Problem Statement

Lisa is studying polynomials in her class. She is learning about the multiplication of polynomials.

To practice her understanding, she wants to write a program that multiplies two polynomials and displays the result. Each polynomial is represented as a linked list, where each node contains the coefficient and exponent of a term.

Example

Input:

4 3

y

3 1

y

1 0

n

2 2

y

3 1

y

2 0

n

Output:

8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Explanation

1. Poly1: 4x^3 + 3x + 1
2. Poly2: 2x^2 + 3x + 2

Multiplication Steps:

1. Multiply 4x^3 by Poly2:
   -> 4x^3 * 2x^2 = 8x^5
   -> 4x^3 * 3x = 12x^4
   -> 4x^3 * 2 = 8x^3
2. Multiply 3x by Poly2:
   -> 3x * 2x^2 = 6x^3
   -> 3x * 3x = 9x^2
   -> 3x * 2 = 6x

3. Multiply 1 by Poly2:

-> 1 * 2x^2 = 2x^2

-> 1 * 3x = 3x

-> 1 * 2 = 2

Combine the results: 8x^5 + 12x^4 + (8x^3 + 6x^3) + (9x^2 + 2x^2) + (6x + 3x) + 2

The combined polynomial is: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Input Format

The input consists of two sets of polynomial terms.

Each polynomial term is represented by two integers separated by a space:

- The first integer represents the coefficient of the term.
- The second integer represents the exponent of the term.

After entering a polynomial term, the user is prompted to input a character indicating whether to continue adding more terms to the polynomial.

If the user inputs 'y' or 'Y', the program continues to accept more terms.

If the user inputs 'n' or 'N', the program moves on to the next polynomial.

Output Format

The output consists of a single line representing the resulting polynomial after multiplying the two input polynomials.

Each term of the resulting polynomial is formatted as follows:

- The coefficient and exponent are separated by 'x^' if the exponent is greater than 1.

- If the exponent is 1, only 'x' is displayed without the exponent.
- If the exponent is 0, only the coefficient is displayed.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 4 3
y
3 1
y
1 0
n
2 2
y
3 1
y
2 0
n
Output: 8x^5 + 12x^4 + 14x^3 + 11x^2 + 9x + 2

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct Node
{
   int coeff;
   int exp;
   struct Node*next;
}Node;

Node*createNode(int coeff,int exp)
{
   Node*newNode = (Node*)malloc(sizeof(Node));
   newNode->coeff = coeff;
   newNode -> exp = exp;
   newNode->next = NULL;
   return newNode;
}
```

```
void insert(Node**poly,int coeff,int exp)
{
   Node*newNode = createNode(coeff,exp);
   if(*poly == NULL)
   {
      *poly = newNode;
   }
   else{
      Node*temp = *poly;
      while(temp -> next!=NULL)
        temp=temp->next;
      temp->next = newNode;
   }
}
Node*multiplyPolynomials(Node*poly1,Node*poly2)
{
   if(!poly1 || !poly2)return NULL;

   Node*result= NULL;
   Node*temp1= poly1;
   Node*temp2;

   while(temp1)
   {
      temp2 = poly2;
      while(temp2)
      {
         int coeff = temp1 -> coeff*temp2->coeff;
         int exp = temp1 -> exp + temp2->exp;
         insert(&result,coeff,exp);
         temp2 = temp2->next;
      }
      temp1 = temp1->next;
   }
   Node *ptr1,*ptr2,*prev;
   ptr1 =result;
   while(ptr1 && ptr1->next)
   {
      prev = ptr1;
      ptr2 = ptr1 -> next;
      while(ptr2)
      {
```

```
            if(ptr1 -> exp == ptr2->exp)
            {
                ptr1 -> coeff += ptr2->coeff;
                prev -> next = ptr2->next;
                free(ptr2);
                ptr2 = prev ->next;
            }
            else {
                prev = ptr2;
                ptr2 = ptr2 ->next;
            }
        }
        ptr1 = ptr1->next;
    }
    return result;
}
void printPolynomial(Node*poly)
{
    int first = 1;
    while(poly)
    {
        if(!first)printf(" + ");
        if(poly->exp == 0)
        {
            printf("%d",poly->coeff);
        }
        else if(poly -> exp == 1)
        {
            printf("%dx",poly->coeff);
        }
        else
        {
            printf("%dx^%d",poly->coeff,poly -> exp);
        }
        first =0;
        poly = poly->next;
    }
    printf("\n");
}
Node*readPolynomials()
{
    Node*poly = NULL;
```

```
    int coeff,exp;
    char choice;

    do{
        scanf("%d",&coeff);
        scanf("%d",&exp);
        insert(&poly,coeff,exp);
        getchar();

        scanf("%c",&choice);
    }while(choice == 'y' || choice =='Y');

    return poly;
}
int main(){
    Node*poly1 = readPolynomials();
    Node*poly2 = readPolynomials();

    Node*result = multiplyPolynomials(poly1,poly2);

    printPolynomial(result);
    return 0;
}
```

Status : Correct

Marks : 10/10

2. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of 13x2: 13 * 12 = 13.

Calculate the value of 12x1: 12 * 11 = 12.

Calculate the value of 11x0: 11 * 10 = 11.

Add the values of x2, x1, and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

The fifth line consists of an integer representing the value of x, at which the polynomial should be evaluated.

Output Format

The output is an integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
13
12
11
1
Output: 36

Answer

```
// You are using GCC
#include<stdio.h>
typedef struct Poly
{
   int a,b,c;
}Polynomial;

int evaluatePolynomial(Polynomial poly , int x)
{
   return(poly.a * x *x)+(poly.b*x)+poly.c;
}
int main()
{
   Polynomial poly;
   int degree,x;


   scanf("%d",&degree);

   scanf("%d",&poly.a);
```

```
    scanf("%d",&poly.b);
    scanf("%d",&poly.c);
    scanf("%d",&x);

    int result = evaluatePolynomial(poly,x);
    printf("%d\n",result);
    return 0;

}
```

Status : Correct ***Marks : 10/10***

3. Problem Statement

Hasini is studying polynomials in her class. Her teacher has introduced a new concept of two polynomials using linked lists.

The teacher provides Hasini with a program that takes two polynomials as input, represented as linked lists, and then displays them together. The polynomials are simplified and should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints the first polynomial.

The second line of output prints the second polynomial.

The polynomials should be displayed in the format ax^b, where a is the coefficient and b is the exponent.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
1 2
2 1
3 0
3
2 2
1 1
4 0

Output: 1x^2 + 2x + 3
2x^2 + 1x + 4

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
typedef struct node
{
    int c;
    int e;
    struct node* next;

}Node;
void insert(Node** list,int co,int ex){
    Node* temp=*list;
    Node* prev= NULL;
    while(temp!=NULL)
    {
        prev=temp;
        temp=temp -> next;
    }
    if(temp!= NULL && temp ->e ==ex){
        temp->c +=co;
        return;
```

```
    }
    Node* newNode = (Node*)malloc(sizeof(Node));
    newNode -> c=co;
    newNode -> e=ex;
    newNode -> next =temp;
    if(prev==NULL){
       *list = newNode;
    }
    else{
       prev -> next = newNode;
    }
}
void print_poly(Node*list)
{
    int f=1;
    Node* temp=list;
    int h=0;
    while(temp != NULL)
    {
       if(temp ->c !=0)
       {
          if(!f)
             printf(" + ");
          if(temp -> e >1)
             printf("%dx^%d",temp -> c,temp -> e);
          else if(temp -> e ==1)
             printf("%dx",temp -> c);
          else
             printf("%d",temp -> c);
          f=0;
          h=1;

       }
       temp = temp -> next;
    }
    if(!h)
          printf("0");
}
int main()
{
    Node* poly1 = NULL;
    Node* poly2 = NULL;
```

```
    int n,m,c,e;
    scanf("%d",&n);
    for(int i=0;i<n;i++)
    {
        scanf("%d %d",&c,&e);
        insert(&poly1,c,e);
    }
    scanf("%d",&m);
    for(int i=0;i<m;i++)
    {
        scanf("%d %d",&c,&e);
        insert(&poly2,c,e);
    }
    print_poly(poly1);
    printf("\n");
    print_poly(poly2);
}
```

Status : Partially correct ***Marks : 3/10***