Rajalakshmi Engineering College

Name: Sherin Katherina
Email: 240701495@rajalakshmi.edu.in
Roll no: 240701495
Phone: 9150930353
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 : 21

Section 1 : Coding

1. 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>
#include<stdlib.h>
#include<math.h>
struct node{
    int coeff;
    struct node*Next;
};
typedef struct node Node;
Node*createNode(int coeff){
    Node*Newnode=(Node*)malloc(sizeof(Node));
    Newnode->coeff=coeff;
    Newnode->Next=NULL;
    return Newnode;
}
int evaluatePoly(Node*List,int x){
    int result=0;
    int power=2;
    Node*temp=List;
    while(temp!=NULL){
        result+=temp->coeff*pow(x,power);
        power--;
        temp=temp->Next;
    }
    return result;
}
int main(){
    int degree,x;
    int coeff0,coeff1,coeff2;
    scanf("%d",&degree);
    scanf("%d",&coeff0);
    scanf("%d",&coeff1);
    scanf("%d",&coeff2);
    scanf("%d",&x);
    Node*List=createNode(coeff2);
    List->Next=createNode(coeff1);
    List->Next->Next=createNode(coeff0);
```

```
    int result=evaluatePoly(List,x);
    printf("%d\n",result);
    free(List->Next->Next);
    free(List->Next);
    free(List);
    return 0;
}
```

Status : Partially correct **Marks : 1/10**

2. Problem Statement

Akila is a tech enthusiast and wants to write a program to add two polynomials. Each polynomial is represented as a linked list, where each node in the list represents a term in the polynomial.

A term in the polynomial is represented in the format ax^b, where a is the coefficient and b is the exponent.

Akila needs your help to implement a program that takes two polynomials as input, adds them, and stores the result in ascending order in a new polynomial-linked list. Write a program to help her.

Input Format

The input consists of lines containing pairs of integers representing the coefficients and exponents of polynomial terms.

Each line represents a single term, with the coefficient and exponent separated by a space.

The input for each polynomial ends with a line containing "0 0".

Output Format

The output consists of three lines representing the first, second, and resulting polynomial after the addition operation, with terms sorted in ascending order of exponents.

Each line contains terms of the polynomial in the format "coefficientx^exponent", separated by " + ".

If the resulting polynomial is zero, the output is "0".

Refer to the sample output for the formatting specifications.

Sample Test Case

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

Output: 1x^2 + 2x^3 + 3x^4
1x^2 + 2x^3 + 3x^4
2x^2 + 4x^3 + 6x^4

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
  int coeff;
  int expo;
  struct node*Next;
};
typedef struct node Node;
Node*createNode(int coeff,int expo){
   Node*Newnode=(Node*)malloc(sizeof(Node));
   Newnode->coeff=coeff;
   Newnode->expo=expo;
   Newnode->Next=NULL;
   return Newnode;
}
Node*Insert(Node*List,int coeff,int expo){
   Node*Newnode=createNode(coeff,expo);
   if(!List||List->expo>expo){
       Newnode->Next=List;
```

```
    return Newnode;
  }
  Node*Position=List;
  while(Position->Next && Position->Next->expo<=expo){
    Position=Position->Next;
  }
  if(Position->expo==expo){
    Position->coeff+=coeff;
    free(Newnode);
  }
  else{
    Newnode->Next=Position->Next;
    Position->Next=Newnode;
  }
  return List;
}
void display(Node*List){
  if(!List){
    printf("0\n");
    return;
  }
  int first=1;
  while(List){
    if(List->coeff!=0){
      if(!first)printf("+");
      {
        printf("%dx^%d",List->coeff,List->expo);
      first=0;
      }
      List=List->Next;
    }
    printf("\n");
  }
}
Node*addPoly(Node*p1,Node*p2){
  Node*result=NULL;
  while(p1&&p2){
    if(p1->expo<p2->expo){
      result=Insert(result,p1->coeff,p1->expo);
      p1=p1->Next;
    }
    else if(p1->expo>p2->expo){
```

```
            result=Insert(result,p2->coeff,p2->expo);
            p2=p2->Next;
        }
        else{
            result=Insert(result,p1->coeff+p2->coeff,p1->expo);
            p1=p1->Next;
            p2=p2->Next;
        }
    }
    while(p1){
        result=Insert(result,p1->coeff,p1->expo);
        p1=p1->Next;
    }
    while(p2){
        result=Insert(result,p2->coeff,p2->expo);
        p2=p2->Next;
    }
    return result;
}
int main()
{
    Node*poly1=NULL;
    Node*poly2=NULL;
    int coeff,expo;
    while(1){
        scanf("%d%d",&coeff,&expo);
        if(coeff==0 && expo==0)break;
        poly1=Insert(poly1,coeff,expo);
    }
    display(poly1);
    while(1){
        scanf("%d%d",&coeff,&expo);
        if(coeff==0 && expo==0)break;
        poly2=Insert(poly2,coeff,expo);
    }
    display(poly2);
    Node*result=addPoly(poly1,poly2);
    display(result);
    return 0;
}
```

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

3. Problem Statement

John is working on a math processing application, and his task is to simplify polynomials entered by users. The polynomial is represented as a linked list, where each node contains two properties:

Coefficient of the term.

Exponent of the term.

John's goal is to combine all the terms that have the same exponent, effectively simplifying the polynomial.

Input Format

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

The next n lines of input consist of two integers, representing the coefficient and exponent of the polynomial in each line separated by space.

Output Format

The first line of output prints the original polynomial in the format 'cx^e + cx^e + ...' (where c is the coefficient and e is the exponent of each term).

The second line of output displays the simplified polynomial in the same format as the original polynomial.

If the polynomial is 0, then only '0' will be printed.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3
5 2
3 1
6 2
Output: Original polynomial: 5x^2 + 3x^1 + 6x^2
Simplified polynomial: 11x^2 + 3x^1

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int coeff;
    int expo;
    struct node*Next;
};
typedef struct node Poly;
void create(Poly*);
void display(Poly*);
void simplify(Poly*poly1,Poly*result);
int main()
{
    int n;
    Poly*poly1=(Poly*)malloc(sizeof(Poly));
    Poly*result=(Poly*)malloc(sizeof(Poly));
    poly1->Next=NULL;
    if(scanf("%d",&n)!=1)return 1;
    for(int i=0;i<n;i++){
        create(poly1);
    }
    printf("Original polynomial:");
    display(poly1);
    simplify(poly1,result);
    printf("Simplified polynomial:");
    display(result);
    return 0;
}
void display(Poly*List){
    Poly*position=List->Next;
    while(position!=NULL){
        printf("%dx^%d ",position->coeff,position->expo);
        position=position->Next;
        if(position!=NULL && position->coeff>0){
            printf("+");
        }
    }
    printf("\n");
}
```

```
void simplify(Poly*poly1,Poly*result)
{
   Poly*pos1=poly1->Next;
   Poly*temp,*rpos=result;
   while(pos1!=NULL)
   {
      temp=result->Next;
      while(temp!=NULL){
         if(temp->expo==pos1->expo){
            temp->coeff+=pos1->coeff;
            break;
         }
         temp=temp->Next;
      }
      if(temp==NULL){
         Poly*Newnode=(Poly*)malloc(sizeof(Poly));
         Newnode->coeff=pos1->coeff;
         Newnode->expo=pos1->expo;
         Newnode->Next=NULL;
         rpos->Next=Newnode;
         rpos=Newnode;
      }
      pos1=pos1->Next;
   }
}
void create(Poly*List){
   Poly*position,*Newnode;
   position=List;
   Newnode=(Poly*)malloc(sizeof(Poly));
   if(scanf("%d %d",&Newnode->coeff,&Newnode->expo)!=2){
      free(Newnode);
      return ;
   }
   Newnode->Next=NULL;
   while(position->Next!=NULL)
   {
      position=position->Next;
   }
   position->Next=Newnode;
}
```

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