Rajalakshmi Engineering College

Name: Sam Devaraja J
Email: 240701463@rajalakshmi.edu.in
Roll no: 2116240701463
Phone: 7395954829
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 : 2
Total Mark : 30
Marks Obtained : 28

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>
struct Node {
    int coefficient;
    int exponent;
    struct Node* next;
};
struct Node* createNode(int coefficient, int exponent) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->coefficient = coefficient;
    newNode->exponent = exponent;
    newNode->next = NULL;
    return newNode;
}
void insertTerm(struct Node** head, int coefficient, int exponent) {
    if (coefficient == 0) {
        return;
```

```
    }
    if (*head == NULL) {
        *head = createNode(coefficient, exponent);
    } else {
        struct Node* temp = *head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
        temp->next = createNode(coefficient, exponent);
    }
}
void displayPolynomial(struct Node* head) {
    struct Node* temp = head;
    int isFirst = 1;
    while (temp != NULL) {
        if (temp->coefficient != 0) {
            if (!isFirst && temp->coefficient > 0) {
                printf(" + ");
            }
            if (temp->exponent == 0) {
                printf("%d", temp->coefficient);
            } else if (temp->exponent == 1) {
                printf("%dx", temp->coefficient);
            } else {
                printf("%dx^%d", temp->coefficient, temp->exponent);
            }
            isFirst = 0;
        }
        temp = temp->next;
    }
    printf("\n");
}
void removeDuplicates(struct Node* head) {
    struct Node* ptr1, *ptr2, *dup;
    ptr1 = head;
    while (ptr1 != NULL && ptr1->next != NULL) {
        ptr2 = ptr1;
        while (ptr2->next != NULL) {
            if (ptr1->exponent == ptr2->next->exponent) {
                ptr1->coefficient = ptr1->coefficient + ptr2->next->coefficient;
                dup = ptr2->next;
                ptr2->next = ptr2->next->next;
```

```
            free(dup);
        } else {
            ptr2 = ptr2->next;
        }
    }
    ptr1 = ptr1->next;
  }
}
struct Node* multiplyPolynomials(struct Node* poly1, struct Node* poly2) {
  struct Node* result = NULL;
  struct Node* temp1 = poly1;
  while (temp1 != NULL) {
    struct Node* temp2 = poly2;
    while (temp2 != NULL) {
      int coeff = temp1->coefficient * temp2->coefficient;
      int exp = temp1->exponent + temp2->exponent;
      insertTerm(&result, coeff, exp);
      temp2 = temp2->next;
    }
    temp1 = temp1->next;
  }
  removeDuplicates(result);
  return result;
}
void freePolynomial(struct Node* head) {
  while (head != NULL) {
    struct Node* temp = head;
    head = head->next;
    free(temp);
  }
}
int main() {
  struct Node* poly1 = NULL;
  struct Node* poly2 = NULL;
  struct Node* result = NULL;
  int coefficient, exponent;
  char choice;
  do {
    scanf("%d %d", &coefficient, &exponent);
    insertTerm(&poly1, coefficient, exponent);
    scanf(" %c", &choice);
  } while (choice == 'y' || choice == 'Y');
```

```
    do {
        scanf("%d %d", &coefficient, &exponent);
        insertTerm(&poly2, coefficient, exponent);
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');
    result = multiplyPolynomials(poly1, poly2);
    displayPolynomial(result);
    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result);
    return 0;
}
```

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

2. 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

```
#include<stdio.h>
#include<stdlib.h>
struct node{
   int coe;
   int pwr;
   struct node *next;
};
typedef struct node Node;
void create(Node* l)
{
   int a;
   scanf("%d\n",&a);
   for(int i=0;i<a;i++)
   {
      Node *n=(Node*)malloc(sizeof(Node));
      n->next=NULL;
      scanf("%d %d",&n->coe,&n->pwr);
      Node*p=l;
```

```
        while(p->next!=NULL)
        p=p->next;
        p->next=n;
    }
}
void display(Node* l)
{
    Node* p=l->next;
    if(p==NULL)
    printf("0x^0");
    while(p!=NULL){
        if(p->pwr==0)
        printf("%d",p->coe);
        else if(p->pwr==1)
        printf("%dx",p->coe);
        else
        printf("%dx^%d",p->coe,p->pwr);
        p=p->next;
        if(p!=NULL && p->coe>0)
        printf(" + ");
    }
    printf("\n");
}
int main()
{
    Node*p1=(Node*)malloc(sizeof(Node));
    Node*p2=(Node*)malloc(sizeof(Node));
    p1->next=NULL;
    p2->next=NULL;
    create(p1);
    create(p2);
    display(p1);
    display(p2);
    return 0;
}
```

Status : Partially correct **Marks : 8/10**

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 coefficient;
    int exponent;
    struct Node* next;
};

struct Node* createNode(int coef, int exp) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->coefficient = coef;
    newNode->exponent = exp;
    newNode->next = NULL;
    return newNode;
}

struct Polynomial {
    struct Node* head;
};

void initPolynomial(struct Polynomial* polynomial) {
    polynomial->head = NULL;
}

void addTerm(struct Polynomial* polynomial, int coef, int exp) {
    struct Node* newNode = createNode(coef, exp);

    if (polynomial->head == NULL) {
        polynomial->head = newNode;
    } else {
        struct Node* ptr = polynomial->head;
        while (ptr->next != NULL) {
            ptr = ptr->next;
        }
        ptr->next = newNode;
    }
}

void displayPolynomial(struct Polynomial* polynomial) {
    if (polynomial->head == NULL) {
        printf("0");
        return;
```

```
    }
    struct Node* current = polynomial->head;
    while (current != NULL) {
        printf("%dx^%d", current->coefficient, current->exponent);
        if (current->next != NULL) {
            printf(" + ");
        }
        current = current->next;
    }
}

void simplifyPolynomial(struct Polynomial* polynomial) {
    if (polynomial->head == NULL || polynomial->head->next == NULL) {
        return;
    }
    struct Node* ptr1 = polynomial->head;
    struct Node* ptr2;
    struct Node* prev = NULL;
    while (ptr1 != NULL) {
        ptr2 = ptr1->next;
        prev = ptr1;
        while (ptr2 != NULL) {
            if (ptr1->exponent == ptr2->exponent) {
                ptr1->coefficient += ptr2->coefficient;
                prev->next = ptr2->next;
                free(ptr2);
                ptr2 = prev->next;
            } else {
                prev = ptr2;
                ptr2 = ptr2->next;
            }
        }
        ptr1 = ptr1->next;
    }
}

int main() {
    struct Polynomial polynomial;
    initPolynomial(&polynomial);
    int terms;
    scanf("%d", &terms);
    for (int i = 0; i < terms; i++) {
```

```
        int coefficient, exponent;
        scanf("%d %d", &coefficient, &exponent);
        addTerm(&polynomial, coefficient, exponent);
    }
    printf("Original polynomial: ");
    displayPolynomial(&polynomial);
    printf("\nSimplified polynomial: ");
    simplifyPolynomial(&polynomial);
    displayPolynomial(&polynomial);
    return 0;
}
```

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