Rajalakshmi Engineering College

Name: Ravi Sankar
Email: 240701424@rajalakshmi.edu.in
Roll no: 240701424
Phone: 8122932671
Branch: REC
Department: I CSE FD
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 : 27

Section 1 : Coding

1. 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 coeff;
    int exp;
    struct Node* next;
};
```

```
// Create a new node
struct Node* createNode(int coeff, int exp) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->coeff = coeff;
   newNode->exp = exp;
   newNode->next = NULL;
   return newNode;
}

// Insert at the end
struct Node* insertTerm(struct Node* head, int coeff, int exp) {
   struct Node* newNode = createNode(coeff, exp);
   if (head == NULL)
      return newNode;

   struct Node* temp = head;
   while (temp->next)
      temp = temp->next;
   temp->next = newNode;
   return head;
}

// Print polynomial
void printPolynomial(struct Node* head) {
   struct Node* temp = head;
   int isFirst = 1;

   while (temp) {
      if (temp->coeff != 0) {
         // Handle sign
         if (!isFirst && temp->coeff > 0)
            printf(" + ");
         else if (!isFirst && temp->coeff < 0)
            printf(" - ");
         else if (isFirst && temp->coeff < 0)
            printf("-");

         // Handle coefficient
         if (!isFirst && temp->coeff < 0)
            printf("%d", -temp->coeff);
         else
            printf("%d", temp->coeff > 0 ? temp->coeff : -temp->coeff);
            // Handle exponent
            if (temp->exp > 1)
                printf("x^%d", temp->exp);
            else if (temp->exp == 1)
                printf("x");

            isFirst = 0;
        }
        temp = temp->next;
    }

    // Handle if all terms were 0
    if (isFirst)
        printf("0");

    printf("\n");
}

int main() {
    int n, m, coeff, exp;
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;

    // First polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        poly1 = insertTerm(poly1, coeff, exp);
    }

    // Second polynomial
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        poly2 = insertTerm(poly2, coeff, exp);
    }

    printPolynomial(poly1);
    printPolynomial(poly2);

    return 0;
```

}

Status : Correct *Marks : 10/10*

2. 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

```
#include <stdio.h>
#include <stdlib.h>

typedef struct Term {
   int coefficient;
   int exponent;
   struct Term *next;
} Term;

Term* createTerm(int coefficient, int exponent) {
   Term* newTerm = (Term*)malloc(sizeof(Term));
   newTerm->coefficient = coefficient;
   newTerm->exponent = exponent;
   newTerm->next = NULL;
   return newTerm;
}

void insertTerm(Term** head, int coefficient, int exponent) {
   Term* newTerm = createTerm(coefficient, exponent);
   if (*head == NULL) {
      *head = newTerm;
   } else {
      Term* temp = *head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newTerm;
   }
}

void printPolynomial(Term* head) {
   Term* temp = head;
   int firstTerm = 1;
   while (temp != NULL) {
      if (!firstTerm) {
         printf(" + ");
```

```
        }
        printf("%dx^%d", temp->coefficient, temp->exponent);
        firstTerm = 0;
        temp = temp->next;
    }
    if (firstTerm) {
        printf("0");
    }
    printf("\n");
}

void simplifyPolynomial(Term** head) {
    Term* current = *head;
    Term* prev = NULL;
    while (current != NULL) {
        Term* runner = current->next;
        prev = current;
        while (runner != NULL) {
            if (runner->exponent == current->exponent) {
                current->coefficient += runner->coefficient;
                prev->next = runner->next;
                free(runner);
                runner = prev->next;
            } else {
                prev = runner;
                runner = runner->next;
            }
        }
        current = current->next;
    }
}

void freePolynomial(Term* head) {
    Term* temp;
    while (head != NULL) {
        temp = head;
        head = head->next;
        free(temp);
    }
}

int main() {
    int n;
    scanf("%d", &n);

    Term* polynomial = NULL;

    for (int i = 0; i < n; i++) {
        int coefficient, exponent;
        scanf("%d %d", &coefficient, &exponent);
        insertTerm(&polynomial, coefficient, exponent);
    }

    printf("Original polynomial: ");
    printPolynomial(polynomial);

    simplifyPolynomial(&polynomial);

    printf("Simplified polynomial: ");
    printPolynomial(polynomial);

    freePolynomial(polynomial);

    return 0;
}
```

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

3. Problem Statement

Rani is studying polynomials in her class. She has learned about polynomial multiplication and is eager to try it out on her own. However, she finds the process of manually multiplying polynomials quite tedious. To make her task easier, she decides to write a program to multiply two polynomials represented as linked lists.

Help Rani by designing a program that takes two polynomials as input and outputs their product polynomial. Each polynomial is represented by a linked list of terms, where each term has a coefficient and an exponent. The terms are entered in descending order of exponents.

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 third line of output prints the resulting polynomial after multiplying the given polynomials.

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

Refer to the sample output for the exact format.

Sample Test Case

Input: 2
2 3
3 2
2
3 2
2 1

Output: 2x^3 + 3x^2
3x^2 + 2x
6x^5 + 13x^4 + 6x^3

Answer

```
#include <stdio.h>
#include <stdlib.h>
```

```
struct Node {
    int coeff;
    int exp;
    struct Node* next;
};

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

struct Node* insertEnd(struct Node* head, int coeff, int exp) {
    struct Node* newNode = createNode(coeff, exp);
    if (!head) return newNode;
    struct Node* temp = head;
    while (temp->next) temp = temp->next;
    temp->next = newNode;
    return head;
}

void displayPoly(struct Node* head) {
    struct Node* temp = head;
    int first = 1;
    while (temp) {
        if (temp->coeff != 0) {
            if (!first) {
                if (temp->coeff > 0) printf(" + ");
                else printf(" - ");
            } else {
                if (temp->coeff < 0) printf("-");
            }

            int absCoeff = temp->coeff > 0 ? temp->coeff : -temp->coeff;

            if (temp->exp == 0) {
                printf("%d", absCoeff);
            } else if (temp->exp == 1) {
                printf("%dx", absCoeff);
            } else {
                printf("%dx^%d", absCoeff, temp->exp);
            }

            first = 0;
        }
        temp = temp->next;
    }
    if (first) printf("0"); // If all coeffs were 0
    printf("\n");
}

struct Node* multiplyPoly(struct Node* p1, struct Node* p2) {
    int terms[2001] = {0};

    for (struct Node* a = p1; a != NULL; a = a->next) {
        for (struct Node* b = p2; b != NULL; b = b->next) {
            terms[a->exp + b->exp] += a->coeff * b->coeff;
        }
    }

    struct Node* result = NULL;
    for (int i = 2000; i >= 0; i--) {
        if (terms[i] != 0)
            result = insertEnd(result, terms[i], i);
    }

    if (!result) result = insertEnd(result, 0, 0); // for all zero
    return result;
}

int main() {
    int n, m, coeff, exp;
    struct Node *poly1 = NULL, *poly2 = NULL;

    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coeff, &exp);
        poly1 = insertEnd(poly1, coeff, exp);
    }

    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coeff, &exp);
        poly2 = insertEnd(poly2, coeff, exp);
    }

    struct Node* product = multiplyPoly(poly1, poly2);

    displayPoly(poly1);
    displayPoly(poly2);
    displayPoly(product);

    return 0;
}
```

Status : Partially correct ***Marks : 7/10***