Rajalakshmi Engineering College

Name: Navaneetha Krishnan
Email: 241901067@rajalakshmi.edu.in
Roll no: 241901067
Phone: 8939010233
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 27.5

Section 1 : Coding

1. Problem Statement

Keerthi is a tech enthusiast and is fascinated by polynomial expressions. She loves to perform various operations on polynomials.

Today, she is working on a program to multiply two polynomials and delete a specific term from the result.

Keerthi needs your help to implement this program. She wants to take the coefficients and exponents of the terms of the two polynomials as input, perform the multiplication, and then allow the user to specify an exponent for deletion from the resulting polynomial, and display the result.

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 representing the coefficient and the exponent of the term in the first polynomial.

The next line 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 representing the coefficient and the exponent of the term in the second polynomial.

The last line consists of an integer, representing the exponent of the term that Keerthi wants to delete from the multiplied polynomial.

Output Format

The first line of output displays the resulting polynomial after multiplication.

The second line displays the resulting polynomial after deleting the specified term.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 3
2 2
3 1
4 0
2
1 2
2 1
2
Output: Result of the multiplication: 2x^4 + 7x^3 + 10x^2 + 8x
Result after deleting the term: 2x^4 + 7x^3 + 8x

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
```

```
typedef struct Node {
    int coef;
    int exp;
    struct Node* next;
} Node;

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

void insertTerm(Node** poly, int coef, int exp) {
    Node* newNode = createNode(coef, exp);
    if (*poly == NULL || (*poly)->exp < exp) {
        newNode->next = *poly;
        *poly = newNode;
    } else {
        Node* temp = *poly;
        while (temp->next != NULL && temp->next->exp > exp) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exp == exp) {
            temp->next->coef += coef;
            free(newNode);
        } else {
            newNode->next = temp->next;
            temp->next = newNode;
        }
    }
}

Node* multiplyPolynomials(Node* poly1, Node* poly2) {
    Node* result = NULL;
    for (Node* p1 = poly1; p1 != NULL; p1 = p1->next) {
        for (Node* p2 = poly2; p2 != NULL; p2 = p2->next) {
            int coef = p1->coef * p2->coef;
            int exp = p1->exp + p2->exp;
            insertTerm(&result, coef, exp);
        }
```

```
    }
    return result;
}

void deleteTerm(Node** poly, int expToDelete) {
    Node* temp = *poly;
    Node* prev = NULL;

    if (temp != NULL && temp->exp == expToDelete) {
        *poly = temp->next;
        free(temp);
        return;
    }

    while (temp != NULL && temp->exp != expToDelete) {
        prev = temp;
        temp = temp->next;
    }

    if (temp != NULL) {
        prev->next = temp->next;
        free(temp);
    }
}

void printPolynomial(Node* poly) {
    if (poly == NULL) {
        printf("0\n");
        return;
    }

    int first = 1;
    for (Node* temp = poly; temp != NULL; temp = temp->next) {
        if (!first && temp->coef > 0) {
            printf(" + ");
        }
        if (temp->coef != 0) {
            if (temp->exp == 0) {
                printf("%d", temp->coef);
            } else if (temp->exp == 1) {
                printf("%dx", temp->coef);
            } else {
```

```
            printf("%dx^%d", temp->coef, temp->exp);
        }
    }
    first = 0;
  }
  printf("\n");
}

int main() {
  int n, m, expToDelete;

  scanf("%d", &n);

  Node* poly1 = NULL;
  for (int i = 0; i < n; i++) {
    int coef, exp;
    scanf("%d %d", &coef, &exp);
    insertTerm(&poly1, coef, exp);
  }


  scanf("%d", &m);

  Node* poly2 = NULL;
  for (int i = 0; i < m; i++) {
    int coef, exp;
    scanf("%d %d", &coef, &exp);
    insertTerm(&poly2, coef, exp);
  }

  scanf("%d", &expToDelete);

  Node* result = multiplyPolynomials(poly1, poly2);

  printf("Result of the multiplication: ");
  printPolynomial(result);

  deleteTerm(&result, expToDelete);

  printf("Result after deleting the term: ");
  printPolynomial(result);
```

```
    return 0;
}
```

Status : Partially correct ***Marks : 7.5/10***

2. Problem Statement

Hayley loves studying polynomials, and she wants to write a program to compare two polynomials represented as linked lists and display whether they are equal or not.

The polynomials are expressed as a series of terms, where each term consists of a coefficient and an exponent. The program should read the polynomials from the user, compare them, and then display whether they are equal or not.

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 representing 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 representing the coefficient and the exponent of the term in the second polynomial.

Output Format

The first line of output prints "Polynomial 1: " followed by the first polynomial.

The second line prints "Polynomial 2: " followed by the second polynomial.

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

If the two polynomials are equal, the third line prints "Polynomials are Equal."

If the two polynomials are not equal, the third line prints "Polynomials are Not Equal."

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 2
1 2
2 1
2
1 2
2 1
Output: Polynomial 1: (1x^2) + (2x^1)
Polynomial 2: (1x^2) + (2x^1)
Polynomials are Equal.

Answer

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>

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

Term* insertTerm(Term* head, int coeff, int exp) {
    Term* newTerm = (Term*)malloc(sizeof(Term));
    if (newTerm == NULL) {
        perror("Memory allocation failed");
        exit(EXIT_FAILURE);
    }
    newTerm->coefficient = coeff;
    newTerm->exponent = exp;
    newTerm->next = NULL;

    if (head == NULL) {
        return newTerm;
    }
```

```
    Term* current = head;
    Term* prev = NULL;

    while (current != NULL && current->exponent > exp) {
        prev = current;
        current = current->next;
    }

    if (current != NULL && current->exponent == exp) {
        current->coefficient += coeff;
        free(newTerm);
        return head;
    }

    if (prev == NULL) {
        newTerm->next = head;
        return newTerm;
    }

    prev->next = newTerm;
    newTerm->next = current;
    return head;
}

int comparePolynomials(Term* poly1, Term* poly2) {
    Term* p1 = poly1;
    Term* p2 = poly2;

    while (p1 != NULL && p2 != NULL) {
        if (p1->coefficient != p2->coefficient || p1->exponent != p2->exponent) {
            return 0;
        }
        p1 = p1->next;
        p2 = p2->next;
    }

    return (p1 == NULL && p2 == NULL);
}

void printPolynomial(Term* poly) {
    Term* current = poly;
    int firstTerm = 1;
```

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

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

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

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

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

    printf("Polynomial 1: ");
    printPolynomial(poly1);
    printf("Polynomial 2: ");
    printPolynomial(poly2);

    if (comparePolynomials(poly1, poly2)) {
```

```
        printf("Polynomials are Equal.\n");
    } else {
        printf("Polynomials are Not Equal.\n");
    }

    freePolynomial(poly1);
    freePolynomial(poly2);

    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>

struct Term {
    int coef;
    int exp;
    struct Term* next;
};

struct Term* createTerm(int coef, int exp) {
    struct Term* newTerm = (struct Term*)malloc(sizeof(struct Term));
    newTerm->coef = coef;
    newTerm->exp = exp;
    newTerm->next = NULL;
    return newTerm;
}

void insertTerm(struct Term** poly, int coef, int exp) {
    struct Term* newTerm = createTerm(coef, exp);
```

```
    if (*poly == NULL || (*poly)->exp < exp) {
        newTerm->next = *poly;
        *poly = newTerm;
    } else {
        struct Term* temp = *poly;
        while (temp->next != NULL && temp->next->exp > exp) {
            temp = temp->next;
        }
        if (temp->next != NULL && temp->next->exp == exp) {
            temp->next->coef += coef;
            free(newTerm);
        } else {
            newTerm->next = temp->next;
            temp->next = newTerm;
        }
    }
}

void printPolynomial(struct Term* poly) {
    if (poly == NULL) {
        printf("0");
        return;
    }

    int firstTerm = 1;
    while (poly != NULL) {
        if (poly->coef != 0) {
            if (!firstTerm) {
                if (poly->coef > 0) {
                    printf(" + ");
                } else {
                    printf(" - ");
                }
            }

            if (firstTerm) {
                firstTerm = 0;
                if (poly->coef < 0) {
                    printf("-");
                }
            }
```

```
            if (poly->exp == 0) {
                printf("%d", abs(poly->coef));
            } else if (poly->exp == 1) {
                printf("%dx", abs(poly->coef));
            } else {
                printf("%dx^%d", abs(poly->coef), poly->exp);
            }
        }
        poly = poly->next;
    }
    printf("\n");
}

int main() {
    int n, m, coef, exp;

    struct Term* poly1 = NULL;
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &coef, &exp);
        insertTerm(&poly1, coef, exp);
    }

    struct Term* poly2 = NULL;
    scanf("%d", &m);
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &coef, &exp);
        insertTerm(&poly2, coef, exp);
    }

    printPolynomial(poly1);
    printPolynomial(poly2);

    return 0;
}
```

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