Rajalakshmi Engineering College

Name: SHRUTI K
Email: 240701510@rajalakshmi.edu.in
Roll no: 240701510
Phone: 9486847824
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

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 data;
   int power;
   struct node *next;
} *head1 = NULL, *head2 = NULL, *temp = NULL, *result = NULL;

struct node* createpoly1(struct node **head, int e, int p) {
   struct node *newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = e;
   newnode->power = p;
   newnode->next = NULL;

   if (*head == NULL) {
      *head = newnode;
   } else {
      temp->next = newnode;
   }
   temp = newnode;  // Update the temp pointer to the new node
   return *head;
}

struct node* multiplypoly(struct node *head1, struct node *head2) {
   struct node *temp1, *temp2, *result = NULL;
   temp1 = head1;

   while (temp1 != NULL) {
      temp2 = head2;
      while (temp2 != NULL) {
         int coef = temp1->data * temp2->data;
         int power = temp1->power + temp2->power;

         // Check if the term already exists in the result
         struct node *temp_result = result;
         int found = 0;

         while (temp_result != NULL) {
            if (temp_result->power == power) {
               temp_result->data += coef;  // Add coefficients for the same power
               found = 1;
               break;
            }
            temp_result = temp_result->next;
         }

         // If the term does not exist, create a new node
         if (!found && coef != 0) {
            result = createpoly1(&result, coef, power);
         }

         temp2 = temp2->next;
      }
      temp1 = temp1->next;
   }
   return result;
}

void display(struct node *head) {
   temp = head;
   int first = 1;  // Flag to manage the formatting of the output

   while (temp != NULL) {
      // Handle formatting for each term
      if (temp->data != 0) {
         if (!first) {
            printf(" + ");
         }
         if (temp->power != 0 && temp->power != 1) {
            printf("%dx^%d", temp->data, temp->power);
         } else if (temp->power == 1) {
            printf("%dx", temp->data);
         } else {
            printf("%d", temp->data);
         }
         first = 0;  // After the first term, set the flag to false
      }
      temp = temp->next;
   }
   printf("\n");
}

int main() {
   int n, e, p;

   // Input the first polynomial
```

```
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &e, &p);
        head1 = createpoly1(&head1, e, p);
    }

    // Input the second polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &e, &p);
        head2 = createpoly1(&head2, e, p);
    }

    // Display the first and second polynomials
    display(head1);
    display(head2);

    // Multiply the two polynomials
    result = multiplypoly(head1, head2);

    // Display the result of the multiplication
    display(result);

    return 0;
}
```

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

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 data;
```

```
    int power;
    struct node *next;
};

// Function prototypes
struct node* createpoly(struct node *head);
void display(struct node *head);

struct node* createpoly(struct node *head) {
    int n;
    struct node *temp = NULL;

    scanf("%d", &n);  // Number of terms
    for (int i = 0; i < n; i++) {
        struct node *newnode = (struct node*)malloc(sizeof(struct node));
        scanf("%d %d", &newnode->data, &newnode->power);
        newnode->next = NULL;

        if (head == NULL) {
            head = newnode;  // First node becomes the head
        } else {
            temp->next = newnode;  // Attach to the last node
        }
        temp = newnode;  // Update temp to the new node
    }
    return head;
}

void display(struct node *head) {
    struct node *temp = head;
    int first = 1;  // Flag to manage formatting (to avoid leading "+" signs)

    while (temp != NULL) {
        if (!first && temp->data > 0) {
            printf(" + ");  // Add "+" for all terms except the first
        }
        if (temp->data != 0) {
            // Handle term formatting based on the power
            if (temp->power != 0 && temp->power != 1) {
                printf("%dx^%d", temp->data, temp->power);
            } else if (temp->power == 1) {
                printf("%dx", temp->data);
            } else {
                printf("%d", temp->data);
            }
            first = 0;  // After the first term, set the flag to false
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    struct node *head1 = NULL, *head2 = NULL;

    // Create the first polynomial
    head1 = createpoly(head1);

    // Create the second polynomial

    head2 = createpoly(head2);

    // Display both polynomials

    display(head1);

    display(head2);

    return 0;
}
```

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

3. 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>

struct node {
   int data;
   int power;
   struct node *next;
} *head1 = NULL, *head2 = NULL, *temp = NULL, *result = NULL, *prev = NULL;

struct node* createpoly1(struct node **head, int e, int p) {
   struct node *newnode = (struct node*)malloc(sizeof(struct node));
   newnode->data = e;
   newnode->power = p;
   newnode->next = NULL;

   if (*head == NULL) {
      *head = newnode;
   } else {
      temp->next = newnode;
   }
   temp = newnode; // Update temp to the new node
   return *head;
}

struct node* multiplypoly(struct node *head1, struct node *head2) {
   struct node *temp1, *temp2, *result = NULL;
   temp1 = head1;
   while (temp1 != NULL) {
      temp2 = head2;
      while (temp2 != NULL) {
         int coef = temp1->data * temp2->data;
         int power = temp1->power + temp2->power;

         struct node *temp_result = result;
         int found = 0;
         while (temp_result != NULL) {
            if (temp_result->power == power) {
               temp_result->data += coef;
```

```
                found = 1;
                break;
            }
            temp_result = temp_result->next;
        }
        if (!found) {
            result = createpoly1(&result, coef, power);
        }
        temp2 = temp2->next;
    }
    temp1 = temp1->next;
}
return result;
}

struct node* deletenode(struct node **result, int s) {
    struct node *temp = *result, *prev = NULL;

    // If the node to be deleted is the first node
    if (temp != NULL && temp->power == s) {
        *result = temp->next; // Move head to next node
        free(temp); // Free the old head
        return *result;
    }

    // Search for the node to delete
    while (temp != NULL && temp->power != s) {
        prev = temp;
        temp = temp->next;
    }

    // If the node wasn't found
    if (temp == NULL) return *result;

    // Unlink the node from the list
    prev->next = temp->next;
    free(temp); // Free the node
    return *result;
}

void display(struct node *head) {
    temp = head;
    while (temp != NULL) {
        if (temp->power != 1 && temp->power != 0) {
            printf("%dx^%d ", temp->data, temp->power);
        }
        if (temp->power == 1) {
            printf("%dx ", temp->data);
        }
        if (temp->power == 0) {
            printf("%d ", temp->data);
        }
        if (temp->next != NULL) {
            printf("+ ");
        }
        temp = temp->next;
    }
    printf("\n");
}

int main() {
    int n, e, p, s;

    // Input the first polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &e, &p);
        head1 = createpoly1(&head1, e, p);
    }

    // Input the second polynomial
    scanf("%d", &n);
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &e, &p);
        head2 = createpoly1(&head2, e, p);
    }

    // Input the power of the term to be deleted
    scanf("%d", &s);

    // Multiply the polynomials
    result = multiplypoly(head1, head2);

    // Display the result of multiplication
    printf("Result of the multiplication: ");
    display(result);

    // Delete the specified term
    result = deletenode(&result, s);

    // Display the result after deletion
    printf("Result after deleting the term: ");
    display(result);

    return 0;
}
```

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