Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 1

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Janani is a tech enthusiast who loves working with polynomials. She wants to create a program that can add polynomial coefficients and provide the sum of their coefficients.

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with 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 output prints the sum of the coefficients of the polynomials.

Sample Test Case

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

Output: 18

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Poly {
   int coeff;
   int expon;
   struct Poly* next;
}Node;
Node* newnode(int coeff, int expon){
   Node* new_node = (Node*) malloc(sizeof(Node));
   new_node->coeff = coeff;
   new_node->expon = expon;
   new_node->next = NULL;
   return new_node;
}
void insertNode(Node** head, int coeff, int expon){
   Node* temp = *head;
   if(temp == NULL) {
      *head = newnode(coeff,expon);
      return;
   }
```

```
    while(temp->next != NULL){
        temp = temp->next;
    }
    temp->next = newnode(coeff,expon);
}
int main()
{
    int n,coeff,expon;
    scanf("%d",&n);
    Node* poly1;
    Node* poly2;
    for(int i=0; i<n; i++)
    {
        scanf("%d %d",&coeff,&expon);
        insertNode(&poly1, coeff, expon);
    }
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d %d",&coeff,&expon);
        insertNode(&poly2, coeff, expon);
    }
    int sum = 0;
    while(poly1 != NULL)
    {
        sum += poly1->coeff;
        poly1 = poly1->next;
    }
    while(poly2 != NULL)
    {
        sum += poly2->coeff;
        poly2 = poly2->next;
    }
    printf("%d",sum);
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Arun is learning about data structures and algorithms. He needs your help in solving a specific problem related to a singly linked list.

Your task is to implement a program to delete a node at a given position. If the position is valid, the program should perform the deletion; otherwise, it should display an appropriate message.

Input Format

The first line of input consists of an integer N, representing the number of elements in the linked list.

The second line consists of N space-separated elements of the linked list.

The third line consists of an integer x, representing the position to delete.

Position starts from 1.

Output Format

The output prints space-separated integers, representing the updated linked list after deleting the element at the given position.

If the position is not valid, print "Invalid position. Deletion not possible."

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
8 2 3 1 7
2
Output: 8 3 1 7

Answer

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

void insert(int);
void display_List();
void deleteNode(int);

struct node {
   int data;
   struct node* next;
} *head = NULL, *tail = NULL;

struct node *newnode,*ptr,*pre=NULL;
void insert(int a)
{newnode=(struct node*)malloc(sizeof(struct node));
newnode->data=a;
newnode->next=NULL;
   if(head==NULL)
   {head=newnode;}
   else
   {tail->next=newnode;}
   tail=newnode;
```

```
}
void display_List()
{ptr=head;
   while(ptr!=NULL)
   {printf("%d ",ptr->data);
      ptr=ptr->next;
   }
}
void deleteNode(int n)
{ptr=head;
if(n==1 && head!=NULL)
{head = head->next;
   display_List();
}
else{
   for(int i=1;i<n && ptr!=NULL;i++)
   {pre=ptr;
   ptr=ptr->next;
   }if(ptr==NULL)
   {printf("Invalid position. Deletion not possible.");

   }
   else
   {pre->next=ptr->next;
   display_List();
   }
}


int main() {
   int num_elements, element, pos_to_delete;

   scanf("%d", &num_elements);

   for (int i = 0; i < num_elements; i++) {
      scanf("%d", &element);
      insert(element);
   }

   scanf("%d", &pos_to_delete);

   deleteNode(pos_to_delete);
```

```
    return 0;
}
```

Status : Wrong ***Marks : 0/10***

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are working on a text processing tool and need to implement a feature that allows users to insert characters at a specific position.

Implement a program that takes user inputs to create a singly linked list of characters and inserts a new character after a given index in the list.

Input Format

The first line of input consists of an integer N, representing the number of characters in the linked list.

The second line consists of a sequence of N characters, representing the linked list.

The third line consists of an integer index, representing the index(0-based) after

which the new character node needs to be inserted.

The fourth line consists of a character value representing the character to be inserted after the given index.

Output Format

If the provided index is out of bounds (larger than the list size):

1. The first line of output prints "Invalid index".
2. The second line prints "Updated list: " followed by the unchanged linked list values.

Otherwise, the output prints "Updated list: " followed by the updated linked list after inserting the new character after the given index.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
a b c d e
2
X
Output: Updated list: a b c X d e

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct Char {
   char value;
   struct Char* next;
}Node;
Node*newnode(char value) {
   Node* new_node = (Node*) malloc(sizeof(Node));
   new_node->value = value;
   new_node->next = NULL;
   return new_node;
}
```

```
void insertNode(Node** head, char value) {
    Node* temp = *head;
    if(temp == NULL) {
        *head = newnode(value);
        return;
    }
    while(temp->next != NULL) {
        temp = temp->next;
    }
    temp->next = newnode(value);
}
int length(Node* head) {
    int len = 0;
    while(head != NULL) {
        head =head->next;
        len++;
    }
    return len;
}
void traverse(Node* head) {
    while(head != NULL) {
        printf("%c ",head->value);
        head = head->next;
    }
    printf("\n");
}
void insert(Node** head, int pos, char value) {
    if(pos >= length(*head)) {
        printf("Invalid index\n");
        return;
    }
    Node* temp = *head;
    for(int i=0; i<pos; i++) {
        temp = temp->next;
    }
    Node* new_node = newnode(value);
    new_node->next = temp->next;
    temp->next = new_node;
}
int main()
{
    int n;
```

```
    char value;
    Node* head = NULL;
    scanf("%d",&n);
    for(int i=0; i<=n; i++) {
        scanf("%c ",&value);
        if(value == ' ' || value == '\n') {
            continue;
        }
        insertNode(&head, value);
    }
    scanf("%d %c",&n,&value);
    insert(&head, n, value);
    printf("Updated list: ");
    traverse(head);
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 4

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

As part of a programming assignment in a data structures course, students are required to create a program to construct a singly linked list by inserting elements at the beginning.

You are an evaluator of the course and guide the students to complete the task.

Input Format

The first line of input consists of an integer N, which is the number of elements.

The second line consists of N space-separated integers.

Output Format

The output prints the singly linked list elements, after inserting them at the beginning.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
78 89 34 51 67

Output: 67 51 34 89 78

Answer

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

struct Node {
   int data;
   struct Node* next;
};

struct Node *head=NULL,*newnode,*ptr;
void insertAtFront(struct Node **head,int a)
{newnode=(struct Node*)malloc(sizeof(struct Node));
newnode->data=a;
newnode->next=NULL;
newnode->next=*head;
   *head=newnode;

}
void printList(struct Node*head)
{ptr=head;
   while(ptr!=NULL)
   {printf("%d ",ptr->data);
   ptr=ptr->next;}
}

int main(){
   struct Node* head = NULL;

   int n;
   scanf("%d", &n);
```

```
    for (int i = 0; i < n; i++) {
        int activity;
        scanf("%d", &activity);
        insertAtFront(&head, activity);
    }

    printList(head);
    struct Node* current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 5

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Imagine you are tasked with developing a simple GPA management system using a singly linked list. The system allows users to input student GPA values, insertion should happen at the front of the linked list, delete record by position, and display the updated list of student GPAs.

Input Format

The first line of input contains an integer n, representing the number of students.

The next n lines contain a single floating-point value representing the GPA of each student.

The last line contains an integer position, indicating the position at which a student record should be deleted. Position starts from 1.

Output Format

After deleting the data in the given position, display the output in the format "GPA: " followed by the GPA value, rounded off to one decimal place.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 4
3.8
3.2
3.5
4.1
2
Output: GPA: 4.1
GPA: 3.2
GPA: 3.8

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct gpa
{
   float value;
   struct gpa* next;
}Node;
Node* newnode(float value)
{
   Node* newgpa = (Node*) malloc(sizeof(Node));
   newgpa->value = value;
   newgpa->next = NULL;
   return newgpa;
}
Node* insertAtStart(Node* head, float value)
{
   Node* newgpa = newnode(value);
   newgpa->next = head;
   return newgpa;
}
```

```
void traverse(Node* head)
{
   while(head != NULL)
   {
      printf("GPA: %.1f\n",head->value);
      head = head->next;
   }
}
void deleteAtPosition(Node** head, int pos)
{
   pos -= 1;
   Node* temp = *head;
   if(pos == 0)
   {
      *head = temp->next;
      free(temp);
      return;
   }
   while(--pos)
   {
      temp = temp->next;
   }
   Node* temp1 = temp->next;
   temp->next = temp->next->next;
   free(temp1);
}
int main()
{
   int n,pos;
   float value;
   scanf("%d",&n);
   Node* head = NULL;
   for(int i=0; i<n; i++)
   {
      scanf("%f",&value);
      head = insertAtStart(head, value);
   }
   scanf("%d",&pos);
   deleteAtPosition(&head, pos);
   traverse(head);
}
```

Status : Correct *Marks : 10/10*

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 6

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

John is tasked with creating a program to manage student roll numbers using a singly linked list.

Write a program for John that accepts students' roll numbers, inserts them at the end of the linked list, and displays the numbers.

Input Format

The first line of input consists of an integer N, representing the number of students.

The second line consists of N space-separated integers, representing the roll numbers of students.

Output Format

The output prints the space-separated integers singly linked list, after inserting the roll numbers of students at the end.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
23 85 47 62 31

Output: 23 85 47 62 31

Answer

```
#include<stdio.h>
#include<stdlib.h>
typedef struct student
{
   int roll;
   struct student* next;
}Node;
Node* newnode(int rollno)
{
   Node* data = (Node*) malloc(sizeof(Node));
   data->roll = rollno;
   data->next = NULL;
   return data;
}
void traverse(Node* head)
{
   while(head != NULL)
   {
      printf("%d ",head->roll);
      head = head->next;
   }
}
int main()
{
   int n,rollno;
   scanf("%d",&n);
   scanf("%d",&rollno);
   Node* head = newnode(rollno);
```

```
    Node* temp = head;
    while(--n)
    {
        scanf("%d",&rollno);
        temp->next = newnode(rollno);
        temp = temp->next;
    }
    traverse(head);
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 7

Attempt : 1
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

Dev is tasked with creating a program that efficiently finds the middle element of a linked list. The program should take user input to populate the linked list by inserting each element into the front of the list and then determining the middle element.

Assist Dev, as he needs to ensure that the middle element is accurately identified from the constructed singly linked list:

If it's an odd-length linked list, return the middle element.If it's an even-length linked list, return the second middle element of the two elements.

Input Format

The first line of input consists of an integer n, representing the number of elements in the linked list.

The second line consists of n space-separated integers, representing the elements of the list.

Output Format

The first line of output displays the linked list after inserting elements at the front.

The second line displays "Middle Element: " followed by the middle element of the linked list.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 5
10 20 30 40 50
Output: 50 40 30 20 10
Middle Element: 30

Answer

```
#include <stdio.h>
#include <stdlib.h>
struct Node {
   int data;
   struct Node* next;
};

struct Node* push(Node* head, int value)
{
   Node* newnode = (struct Node*) malloc(sizeof(struct Node));
   newnode->next = head;
   newnode->data = value;
   return newnode;
}
int printMiddle(struct Node* head)
{
   int len = 0;
   Node* temp = head;
```

```
    while(temp != NULL)
    {
        len++;
        temp = temp->next;
    }
    int pos = len/2;
    for(int i=0; i<pos; i++)
    {
        head = head->next;
    }
    return head->data;
}

int main() {
    struct Node* head = NULL;
    int n;

    scanf("%d", &n);
    int value;

    for (int i = 0; i < n; i++) {
        scanf("%d", &value);
        head = push(head, value);
    }

    struct Node* current = head;
    while (current != NULL) {
        printf("%d ", current->data);
        current = current->next;
    }
    printf("\n");


    int middle_element = printMiddle(head);
    printf("Middle Element: %d\n", middle_element);


    current = head;
    while (current != NULL) {
        struct Node* temp = current;
        current = current->next;
        free(temp);
```

```
    }

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Hruthika Lakshmi K
Email: 240801116@rajalakshmi.edu.in
Roll no: 240801116
Phone: 9445752530
Branch: REC
Department: I ECE AE
Batch: 2028
Degree: B.E - ECE

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 0

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;
};
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;
}
void insertNode(struct Node** head,int coeff,int exp)
{
    struct Node* newNode = createNode(coeff,exp);
    if(*head == NULL)
    {
        *head = newNode;
    }
    else
    {
        struct Node* temp = *head;
        struct Node* prev = NULL;
        while(temp != NULL && temp->exp<exp)
        {
            prev = temp;
        }
        if(temp != NULL && temp->exp == exp)
        {
            temp->coeff += coeff;
            free(newNode);
        }
        else
        {
            if(prev == NULL)
            {
                newNode->next = *head;
                *head = newNode;
            }
            else
            {
                newNode->next = prev->next;
                prev->next = newNode;
            }
        }
    }
}
 struct Node* addPolynomials(struct Node* poly1,struct Node* poly2)
```

```
{
    struct Node* result = NULL;
    while(poly1 != NULL)
    {
        insertNode(&result,poly1->coeff,poly1->exp);
        poly1 = poly1->next;
    }
    while(poly2 != NULL)
    {
        insertNode(&result,poly2->coeff,poly2->exp);
        poly2 = poly2->next;
    }
    return result;
}
void printPolynomials(struct Node* head)
{
    if(head == NULL)
    {
        printf("0\n");
        return;
    }
    struct Node* prev = NULL;
    struct Node* current = head;
    while(current != NULL)
    {
        if(current->coeff == 0)
        {
            if(prev == NULL)
            {
                head = current->next;
                free(current);
                current = head;
            }
            else
            {
                prev->next = current->next;
                free(current);
                current = prev->next;
            }
        }
        else
        {
```

```
            prev = current;
            current = current->next;
        }
    }
    if(head == NULL)
    {
        printf("0\n");
        return;
    }
    while(head != NULL)
    {
        printf("%dx^%d",head->coeff,head->exp);
        head = head->next;
        if(head != NULL)
        {
            printf("+");
        }
    }
    printf("\n");
}
int main()
{
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;
    int coeff;
    int exp;
    while(1)
    {
        scanf("%d %d",&coeff,&exp);
        if(coeff == 0 && exp == 0)break;
        insertNode(&poly1,coeff,exp);
    }
    while(1)
    {
        scanf("%d %d",&coeff,&exp);
        if(coeff == 0 && exp == 0)break;
        insertNode(&poly2,coeff,exp);
    }
    struct Node* result = addPolynomials(poly1,poly2);
    printPolynomials(poly1);
    printPolynomials(poly2);
    printPolynomials(result);
```

```
    return 0;
}
```

Status : Wrong ***Marks : 0/10***

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

```
#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;
}
void insertNode(struct Node**head,int coeff,int exp)
{
   struct Node* newNode = createNode(coeff,exp);
   if(*head == NULL)
   {
      *head = newNode;
   }
   else
   {
```

```
        struct Node* temp = *head;
        while(temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
void printPolynomial(struct Node* head)
{
    if(head == NULL)
    {
        printf("0\n");
        return;
    }
    while(head != NULL)
    {
        if(head->exp == 0)
        {
            printf("%d",head->coeff);
        }
        else if(head->exp == 1)
        {
            printf("%dx^%d",head->coeff,head->exp);
        }
        head = head->next;
        if(head != NULL)
        {
            printf("+");
        }
    }
    printf("\n");
}
int main()
{
    int n,m,coeff,exp;
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;
    scanf("%d",&n);
    for(int i=0; i<n; i++)
    {
        scanf("%d %d",&coeff,&exp);
```

```
        insertNode(&poly1,coeff,exp);
    }
    scanf("%d",&m);
    for(int i=0; i<m; i++)
    {
        scanf("%d %d",&coeff,&exp);
        insertNode(&poly2,coeff,exp);
    }
    printPolynomial(poly1);
    printPolynomial(poly2);
    return 0;
}
```

Status : Wrong **Marks : 0/10**

3. 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

```
#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;
}
void insertNode(struct Node** head,int coeff,int exp)
{
   struct Node* newNode = createNode(coeff,exp);
   if(*head == NULL)
   {
      *head = newNode;
   }
   else
   {
      struct Node* temp = *head;
```

```
        while(temp->next!=NULL)
        {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}
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->coeff* temp2->coeff;
            int exp = temp1->exp + temp2->exp;
            insertNode(&result,coeff,exp);
            temp2 = temp2->next;
        }
        temp1 = temp1->next;
    }
    struct Node* temp = result;
    while(temp != NULL)
    {
        struct Node* temp2 = temp->next;
        while(temp2 != NULL)
        {
            if(temp->exp == temp2->exp)
            {
                temp->coeff += temp2->coeff;
                temp2->coeff = 0;
            }
            temp2 = temp2->next;
        }
        temp = temp->next;
    }
    struct Node* prev = NULL;
    temp = result;
    while(temp != NULL)
    {
```

```
        if(temp->coeff == 0)
        {
            if(prev == NULL)
            {
                result = temp->next;
                free(temp);
                temp = result;
            }
            else
            {
                prev->next = temp->next;
                free(temp);
                temp = prev->next;
            }
        }
        else
        {
            prev = temp;
            temp = temp->next;
        }
    }
    return result;
}
void printPolynomial(struct Node* head)
{
    if(head == NULL)
    {
        printf("0\n");
        return;
    }
    while(head != NULL)
    {
        if(head->exp == 0)
        {
            printf("%d",head->coeff);
        }
        else if(head->exp == 1)
        {
            printf("%dx",head->coeff);
        }
        else
        {
```

```
            printf("%dx^%d",head->coeff,head->exp);
        }
        head = head->next;
        if(head != NULL)
        {
            printf("+");
        }
    }
    printf("\n");
}
int main()
{
    struct Node* poly1 = NULL;
    struct Node* poly2 = NULL;
    int coeff,exp;
    char ch;
    while(1)
    {
        scanf("%d %d",&coeff,&exp);
        insertNode(&poly1,coeff,exp);
        scanf("%c",&ch);
        if(ch == 'n' || ch == 'N')break;
    }
    while(1)
    {
        scanf("%d %d",&coeff,&exp);
        insertNode(&poly2,coeff,exp);
        scanf("%c",&ch);
        if(ch == 'n' || ch == 'N')break;
    }
    struct Node* result = multiplyPolynomials(poly1,poly2);
    printPolynomial(result);
    return 0;
}
```

Status : Wrong **Marks : 0/10**