Rajalakshmi Engineering College

Name: Mukesh V
Email: 240701339@rajalakshmi.edu.in
Roll no: 240701339
Phone: 9597888573
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_MCQ

Attempt : 1
Total Mark : 10
Marks Obtained : 1

Section 1 : MCQ

1. Linked lists are not suitable for the implementation of?

Answer

Binary search

Status : Correct ***Marks : 1/1***

2. Which of the following statements is used to create a new node in a singly linked list?

```
struct node {
   int data;
   struct node * next;
}
typedef struct node NODE;
```

NODE *ptr;

Answer

Status : Skipped ***Marks : 0/1***

3. In a singly linked list, what is the role of the "tail" node?

Answer

Status : Skipped ***Marks : 0/1***

4. The following function takes a singly linked list of integers as a parameter and rearranges the elements of the lists.

The function is called with the list containing the integers 1, 2, 3, 4, 5, 6, 7 in the given order. What will be the contents of the list after the function completes execution?

```
struct node {
   int value;
   struct node* next;
};

void rearrange (struct node* list) {
   struct node *p,q;
   int temp;
   if (! List || ! list->next) return;
   p=list; q=list->next;
   while(q) {
      temp=p->value; p->value=q->value;
      q->value=temp;p=q->next;
      q=p?p->next:0;
   }
}
```

Answer

-

Status : *-* ***Marks : 0/1***

5. Given the linked list: 5 -> 10 -> 15 -> 20 -> 25 -> NULL. What will be the output of traversing the list and printing each node's data?

Answer

-

Status : *-* ***Marks : 0/1***

6. Given a pointer to a node X in a singly linked list. If only one point is given and a pointer to the head node is not given, can we delete node X from the given linked list?

Answer

-

Status : *-* ***Marks : 0/1***

7. The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function.

What should be added in place of "/*ADD A STATEMENT HERE*/", so that the function correctly reverses a linked list?

```
struct node {
   int data;
   struct node* next;
};
static void reverse(struct node** head_ref) {
   struct node* prev   = NULL;
   struct node* current = *head_ref;
   struct node* next;
   while (current != NULL) {
      next  = current->next;
      current->next = prev;
      prev = current;
```

```
    current = next;
  }
  /*ADD A STATEMENT HERE*/
}
```

Answer

-

Status : *-* ***Marks : 0/1***

8. Consider an implementation of an unsorted singly linked list. Suppose it has its representation with a head pointer only. Given the representation, which of the following operations can be implemented in O(1) time?

i) Insertion at the front of the linked list

ii) Insertion at the end of the linked list

iii) Deletion of the front node of the linked list

iv) Deletion of the last node of the linked list

Answer

-

Status : *-* ***Marks : 0/1***

9. Consider the singly linked list: 15 -> 16 -> 6 -> 7 -> 17. You need to delete all nodes from the list which are prime.

What will be the final linked list after the deletion?

Answer

-

Status : *-* ***Marks : 0/1***

10. Consider the singly linked list: 13 -> 4 -> 16 -> 9 -> 22 -> 45 -> 5 -> 16 -> 6, and an integer K = 10, you need to delete all nodes from the list that are

less than the given integer K.

What will be the final linked list after the deletion?

Answer

-

Status : *-* ***Marks : 0/1***

Rajalakshmi Engineering College

Name: Mukesh V
Email: 240701339@rajalakshmi.edu.in
Roll no: 240701339
Phone: 9597888573
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_COD_Question 1

Attempt : 1
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>
struct node{
   int coeff;
   int expo;
   struct node *link;
};

struct node *create_node(int coeff,int expo){
   struct node *temp=(struct node*)malloc(sizeof(struct node));
   temp->coeff=coeff;
   temp->expo=expo;
   temp->link=NULL;
   return temp;

}

void insert(struct node **poly,int coeff,int expo){
   struct node *newnode=create_node(coeff,expo);
   if(*poly==NULL){
```

```
        *poly=newnode;
    }
    else{
        struct node *ptr;
        ptr=*poly;
        while(ptr->link != NULL){
            ptr=ptr->link;
        }
        ptr->link=newnode;
    }
}

int total(struct node **poly){
    int sum=0;
    struct node *ptr;
    ptr=*poly;
    while(ptr!=NULL){
        sum+=ptr->coeff;
        ptr=ptr->link;
    }
    return sum;
}

void free_polynomial(struct node *poly){
    struct node *temp;
    while(poly != NULL){
        temp=poly;
        poly=poly->link;
        free(temp);
    }
}

int main(){
    int sum1,sum2;
    int n,m,coeff,expo;
    struct node *poly1=NULL;
    struct node *poly2=NULL;
    scanf("%d",&n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coeff,&expo);
        insert(&poly1,coeff,expo);
    }
```

```
    scanf("%d",&m);
    for(int i=0;i<m;i++){
        scanf("%d %d",&coeff,&expo);
        insert(&poly1,coeff,expo);
    }
    sum1=total(&poly1);
    sum2=total(&poly2);
    printf("%d",sum1 + sum2);
    free_polynomial(poly1);
    free_polynomial(poly2);



}
```

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

Rajalakshmi Engineering College

Name: Mukesh V
Email: 240701339@rajalakshmi.edu.in
Roll no: 240701339
Phone: 9597888573
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_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 10

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;

void deleteNode(int pos) {
   if (pos <= 0) {
      printf("Invalid position. Deletion not possible.");
      return;
   }

   struct node* temp = head;
   struct node* prev = NULL;
   int i;
```

```
    for (i = 1; i < pos && temp != NULL; i++) {
        prev = temp;
        temp = temp->next;
    }

    if (temp == NULL) {
        printf("Invalid position. Deletion not possible.");
        return;
    }

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

    display_List();
    return;
}

void insert(int value) {
    struct node* newnode;
    newnode = (struct node*)malloc(sizeof(struct node));
    newnode->data = value;
    newnode->next = NULL;

    if (head == NULL) {
        head = newnode;
        tail = newnode;
    } else {
        tail->next = newnode;
        tail = newnode;
    }
    return;
}

void display_List() {
    struct node* temp;
    temp = head;
    while (temp != NULL) {
```

```
        if (temp->next == NULL) {
            printf("%d ", temp->data);
        } else {
            printf("%d ", temp->data);
        }
        temp = temp->next;
    }
    return;
}

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 : Correct **Marks : 10/10**

Rajalakshmi Engineering College

Name: Mukesh V
Email: 240701339@rajalakshmi.edu.in
Roll no: 240701339
Phone: 9597888573
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 : 30

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

```
#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 insert(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;
            insert(&result, coeff, exp);
            temp2 = temp2->next;
        }
        temp1 = temp1->next;
    }

    removeduplicates(result);
    return result;
}
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);
        insert(&poly1, coefficient, exponent);
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');

    do {
        scanf("%d %d", &coefficient, &exponent);
        insert(&poly2, coefficient, exponent);
        scanf(" %c", &choice);
    } while (choice == 'y' || choice == 'Y');


    result = multiplypolynomials(poly1, poly2);

    displaypolynomial(result);

    return 0;
}
```

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

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

```
#include <stdio.h>
#include <stdlib.h>
struct Term {
    int coefficient;
    int exponent;
    struct Term* next;
```

```
};

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

int comparePolynomials(struct Term* poly1, struct Term* poly2) {
    while (poly1 != NULL && poly2 != NULL) {
        if (poly1->coefficient != poly2->coefficient || poly1->exponent != poly2-
>exponent) {
            return 0;
        }
        poly1 = poly1->next;
        poly2 = poly2->next;
    }

    return poly1 == NULL && poly2 == NULL;
}

void displayPolynomial(struct Term* poly) {
    while (poly != NULL) {
        printf("(%dx^%d)", poly->coefficient, poly->exponent);
        poly = poly->next;
        if (poly != NULL) {
            printf(" + ");
        }
    }
    printf("\n");
}

int main() {
    struct Term* poly1 = NULL;
    struct Term* poly2 = NULL;

    int numTerms1, numTerms2;
    int coeff, exp;

    scanf("%d", &numTerms1);
```

```
for (int i = 0; i < numTerms1; i++) {
    scanf("%d %d", &coeff, &exp);
    if (coeff != 0) {
        if (poly1 == NULL) {
            poly1 = createTerm(coeff, exp);
        } else {
            struct Term* temp = poly1;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = createTerm(coeff, exp);
        }
    }
}

scanf("%d", &numTerms2);
for (int i = 0; i < numTerms2; i++) {
    scanf("%d %d", &coeff, &exp);
    if (coeff != 0) {
        if (poly2 == NULL) {
            poly2 = createTerm(coeff, exp);
        } else {
            struct Term* temp = poly2;
            while (temp->next != NULL) {
                temp = temp->next;
            }
            temp->next = createTerm(coeff, exp);
        }
    }
}

printf("Polynomial 1: ");
displayPolynomial(poly1);

printf("Polynomial 2: ");
displayPolynomial(poly2);

if (comparePolynomials(poly1, poly2)) {
    printf("Polynomials are Equal.");
} else {
    printf("Polynomials are Not Equal.");
}
```

```
    return 0;
}
```

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

3. Problem Statement

Timothy wants to evaluate polynomial expressions for his mathematics homework. He needs a program that allows him to input the coefficients of a polynomial based on its degree and compute the polynomial's value for a given input of x. Implement a function that takes the degree, coefficients, and the value of x, and returns the evaluated result of the polynomial.

Example

Input:

degree of the polynomial = 2

coefficient of x2 = 13

coefficient of x1 = 12

coefficient of x0 = 11

x = 1

Output:

36

Explanation:

Calculate the value of 13x2: 13 * 12 = 13.

Calculate the value of 12x1: 12 * 11 = 12.

Calculate the value of 11x0: 11 * 10 = 11.

Add the values of x2, x1, and x0 together: 13 + 12 + 11 = 36.

Input Format

The first line of input consists of an integer representing the degree of the polynomial.

The second line consists of an integer representing the coefficient of x2.

The third line consists of an integer representing the coefficient of x1.

The fourth line consists of an integer representing the coefficient of x0.

The fifth line consists of an integer representing the value of x, at which the polynomial should be evaluated.

Output Format

The output is an integer value obtained by evaluating the polynomial at the given value of x.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 2
13
12
11
1

Output: 36

Answer

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
typedef struct polyTerm {
    int coeff;
    int exp;
    struct polyTerm *next;
} PolyTerm;
PolyTerm *createTerm(int coeff, int exp) {
    PolyTerm *term = (PolyTerm*) malloc(sizeof(PolyTerm));
    term->coeff = coeff;
    term->exp = exp;
    term->next = NULL;
    return term;
```

```
}
void addTerm(PolyTerm **poly, int coeff, int exp) {
    PolyTerm *term = createTerm(coeff, exp);
    if (*poly == NULL) {
        *poly = term;
        return;
    }
    PolyTerm *last = *poly;
    while (last->next != NULL) {
        last = last->next;
    }
    last->next = term;
}
int evaluatePoly(PolyTerm *poly, int x) {
    int result = 0;
    while (poly != NULL) {
        result += poly->coeff * pow(x, poly->exp);
        poly = poly->next;
    }
    return result;
}

int main() {
    int degree, coeff, exp, x;
    PolyTerm *poly = NULL;
    scanf("%d", &degree);

    for (int i = degree; i >= 0; i--) {
        scanf("%d", &coeff);
        if (coeff != 0) {
            addTerm(&poly, coeff, i);
        }
    }
    scanf("%d", &x);
    int result = evaluatePoly(poly, x);
    printf("%d\n", result);
    return 0;
}
```

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