

Rajalakshmi Engineering College

Name: Libin Mathew

Email: 241801137@rajalakshmi.edu.in

Roll no: 241801137

Phone: 9947119753

Branch: REC

Department: I AI & DS AF

Batch: 2028

Degree: B.E - AI & DS

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_MCQ

Attempt : 1

Total Mark : 10

Marks Obtained : 10

Section 1 : MCQ

1. 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

*head_ref = prev;

Status : Correct

Marks : 1/1

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

Answer

It stores the last element of the list

Status : Correct

Marks : 1/1

3. 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

13 -> 16 -> 22 -> 45 -> 16

Status : Correct

Marks : 1/1

4. 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

I and III

Status : Correct

Marks : 1/1

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

Answer

Binary search

Status : Correct

Marks : 1/1

6. 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

2, 1, 4, 3, 6, 5, 7

Status : Correct

Marks : 1/1

7. 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

Possible if X is not last node.

Status : Correct

Marks : 1/1

8. 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

5 10 15 20 25

Status : Correct

Marks : 1/1

9. 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

ptr = (NODE*)malloc(sizeof(NODE));

Status : Correct

Marks : 1/1

10. 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

15 -> 16 -> 6

Status : Correct

Marks : 1/1

Rajalakshmi Engineering College

Name: Libin Mathew
Email: 241801137@rajalakshmi.edu.in
Roll no: 241801137
Phone: 9947119753
Branch: REC
Department: I AI & DS AF
Batch: 2028
Degree: B.E - AI & DS

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;
```

```
// You are using GCC
```

```
typedef struct node Node;
```

```
void insert(int x)
```

```
{
```

```
    Node *newnode;
```

```
    newnode=(Node *)malloc(sizeof(Node));
```

```
    newnode->data=x;
```

```
    newnode->next=NULL;
```

```
    if(head==NULL){
```

```
    head=newnode;
    tail=newnode;
}
else{
    tail->next=newnode;
    tail=newnode;
}
}
```

```
void deleteNode(int pos)
```

```
{
    if(head==NULL){
        tail=NULL;
        printf("Invalid position. Deletion not possible.");
        return;
    }
    if(pos<1){
        printf("Invalid position. Deletion not possible.");
        return;
    }
    struct node *temp=NULL;
    if(pos==1){
        temp=head;
        head=head->next;
        free(temp);
        temp=NULL;
        display_List();
        return;
    }
    int count=1;
    struct node *current=head;
```

```
while(current!=NULL && count<pos)
```

```
{
    temp=current;
    current=current->next;
    count++;
}
```

```
if(current==NULL)
```

```
{
    printf("Invalid position. Deletion not possible.");
```



```

        return;
    }
    else if(current!=NULL){
        temp->next=current->next;
        free(current);
        current=NULL;

        if(temp->next==NULL){
            tail=temp;
        }
        display_List();

        return;
    }
}

void display_List(){
    struct node *current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->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: Libin Mathew

Email: 241801137@rajalakshmi.edu.in

Roll no: 241801137

Phone: 9947119753

Branch: REC

Department: I AI & DS AF

Batch: 2028

Degree: B.E - AI & DS

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1

Total Mark : 30

Marks Obtained : 11

Section 1 : Coding

1. 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 x^2 = 13

coefficient of x^1 = 12

coefficient of $x_0 = 11$

$x = 1$

Output:

36

Explanation:

Calculate the value of $13x^2$: $13 * 12 = 13$.

Calculate the value of $12x^1$: $12 * 11 = 12$.

Calculate the value of $11x^0$: $11 * 10 = 11$.

Add the values of x^2 , x^1 , and x^0 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 x^2 .

The third line consists of an integer representing the coefficient of x^1 .

The fourth line consists of an integer representing the coefficient of x^0 .

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

// You are using GCC

```
#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;
```

```
}
```

```
long long evaluatePolynomial(struct Node* head, int x){
```

```
    long long result = 0;
```

```
    while (head != NULL){
```

```
        long long term = head->coeff;
```

```
        int exponent = head->exp;
```

```
        long long power = 1;
```

```
        for(int i=0;i<exponent;i++){
```

```
            power*=x;
```

```
        }
```

```
        result +=term * power;
```

```
        head = head->next;
```

```
    }
```

```
    return result;
```

```
}
```

```
int main(){
```

```
    int degree,coeff,x;
```

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

```
    struct Node* head = NULL;
```

```

struct Node* temp = NULL;
for(int i = degree; i >=0; i--)
{
    scanf("%d",&coeff);
    struct Node* newNode = createNode(coeff, i);

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

scanf("%d",&x);
printf("%lld\n",evaluatePolynomial(head, x));

while(head != NULL){
    struct Node* temp = head;
    head = head->next;
    free(temp);
}
return 0;
}

```

Status : Correct

Marks : 10/10

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

// You are using GCC

```

#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* addPolynomials(struct Node* poly1, struct Node* poly2){
    struct Node* result = NULL;
    while (poly1 != NULL || poly2 != NULL){
        int coeff, exp;

        if (poly1 == NULL){
            coeff = poly2->coeff;
            exp = poly2->exp;
            poly2 = poly2->next;

        }else if (poly2 == NULL){
            coeff = poly1->coeff;
            exp = poly1->exp;
            poly1 = poly1->next;

```



```

    }else if (poly1->exp > poly2->exp){
        coeff = poly1->coeff;
        exp = poly1->exp;
        poly1 = poly1->next;

    }else if (poly1->exp < poly2->exp){
        coeff = poly2->coeff;
        exp = poly2->exp;
        poly2 = poly2->next;
    }else {
        coeff = poly1->coeff + poly2->coeff;
        exp = poly1->exp;
        poly1 = poly1->next;
        poly2 = poly2->next;
    }
    if (coeff != 0){
        insertNode(&result, coeff, exp);
    }
}
return result;
}

```

```

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

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

    while(1){
        scanf("%d %d",&coeff,&exp);
        if (coeff == 0 && exp == 0)break;
        insertNode(&poly1,coeff,exp);
    }
    freePolynomial(poly1);
    while(1){
        scanf("%d %d",&coeff,&exp);

```

```

        if (coeff == 0 && exp == 0) break;
        insertNode(&poly2, coeff, exp);
    }
    freePolynomial(poly2);

    result = addPolynomials(poly1, poly2);
    printPolynomial(result);

    freePolynomial(poly1);
    freePolynomial(poly2);
    freePolynomial(result);
    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:

$$\rightarrow 4x^3 * 2x^2 = 8x^5$$

$$\rightarrow 4x^3 * 3x = 12x^4$$

$$\rightarrow 4x^3 * 2 = 8x^3$$

2. Multiply $3x$ by Poly2:

$$\rightarrow 3x * 2x^2 = 6x^3$$

$$\rightarrow 3x * 3x = 9x^2$$

$$\rightarrow 3x * 2 = 6x$$

3. Multiply 1 by Poly2:

$$\rightarrow 1 * 2x^2 = 2x^2$$

$$\rightarrow 1 * 3x = 3x$$

$$\rightarrow 1 * 2 = 2$$

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

// You are using GCC

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

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

```
typedef struct Node{
```

```
    int coeff;
```

```
    int exp;
```

```
    struct Node* next;
```

```
}Node;
```

```
Node* createNode(int coeff, int exp)
```

```
{
```

```
    Node* newNode =
```

```
    (Node*)malloc(sizeof(Node));
```

```
    newNode->coeff = coeff;
```

```
    newNode->exp = exp;
```

```
    newNode->next = NULL;
```

```
    return newNode;
```

```
}
```

```
void insertTerm(Node** poly, int coeff, int exp){
```

```
if(coeff == 0)return;
```

```
Node* temp = *poly;
```

```
Node* prev = NULL;
```

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

```
if (temp != NULL && temp->exp==exp){  
    prev = temp;  
    temp = temp->next;  
}
```

```
if(temp != NULL && temp->exp == exp){  
    temp->coeff += coeff;
```

```
}else{  
    Node* newNode = createNode(coeff, exp);  
    if(prev == NULL){  
        newNode->next = *poly;  
        *poly = newNode;  
    }else{  
        newNode->next = temp;  
        prev->next = newNode;  
    }  
}
```

```
Node* multiplyPoly(Node* poly1, Node*poly2){
```

```
    Node* result = NULL;
```

```
    Node* ptr1 = poly1;
```

```
    while (ptr1 != NULL){  
        Node* ptr2 = poly2;  
        while (ptr2 != NULL){  
            int coeff = ptr1->coeff *ptr2->coeff;  
            int exp = ptr1->exp + ptr2->exp;  
            insertTerm(&result, coeff, exp);  
            ptr2 = ptr2->next;  
        }  
        ptr1 = ptr1->next;  
    }
```

```

    ptr1 = ptr1->next;
}
return result;
}

void printPoly(Node* poly){
    while (poly != NULL){
        printf("%dx^%d", poly->coeff, poly->exp);
        poly = poly->next;
        if (poly != NULL)
            printf(" + ");
    }
    printf("\n");
}

Node* readPolynomial(){
    Node* poly = NULL;
    char ch;
    int coeff, exp;
    do{
        scanf("%d %d", &coeff, &exp);
        insertTerm(&poly, coeff, exp);
        scanf(" %c", &ch);
    }while (ch == 'y' || ch == 'Y');
    return poly;
}

int main()
{
    Node* poly1 = NULL;
    Node* poly2 = NULL;

    poly1 = readPolynomial();
    poly2 = readPolynomial();

    Node* result = multiplyPoly(poly1, poly2);

    printPoly(result);
    return 0;
}

```

Status : Partially correct

Marks : 1/10

Rajalakshmi Engineering College

Name: Libin Mathew

Email: 241801137@rajalakshmi.edu.in

Roll no: 241801137

Phone: 9947119753

Branch: REC

Department: I AI & DS AF

Batch: 2028

Degree: B.E - AI & DS

Scan to verify results



NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_PAH_modified

Attempt : 1

Total Mark : 5

Marks Obtained : 0

Section 1 : Coding

1. Problem Statement

Emily is developing a program to manage a singly linked list. The program should allow users to perform various operations on the linked list, such as inserting elements at the beginning or end, deleting elements from the beginning or end, inserting before or after a specific value, and deleting elements before or after a specific value. After each operation, the updated linked list should be displayed.

Your task is to help Emily in implementing the same.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated

integers, with -1 indicating the end of input.

- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an integer value representing the node after which deletion occurs.
- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1

5

3

7

-1

2

11

Output: LINKED LIST CREATED

5 3 7

Answer

// You are using GCC

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node*next;
```

```
};
```

```
struct node*ptr,*newnode,*head=NULL,*last=NULL;
```

```
int check(int val)
```

```
{
```

```
    ptr=head;
```

```
    while(ptr!=NULL)
```

```
    {
```

```
        if(val==ptr->data)
```

```
            return 1;
```

```
        ptr=ptr->next;
```

```
    }
```

```
    return 0;
```

```
}
```

```
int main()
```

```
{
```

```
    int ch;
```

```
    while(1)
```

```
    {
```

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

```
        if(ch==1)
```

```
        {
```

```
            int a=0;
```

```
while(a!=1)
{
    newnode->data=a;
    newnode->next=NULL;
    if(head==NULL)
    {head=newnode;}
    else
    {
        last->next=newnode;
    }
    last=newnode;
}
printf("LINKED LIST CREATED\n");
}
else if(ch==2)
{
    ptr=head;
    if(ptr==NULL)
    {
        printf("The list is empty\n");
    }
    else
    {
        while(ptr!=NULL)
        {
            printf("%d",ptr->data);
            ptr=ptr->next;
        }
        printf("\n");
    }
}
else if(ch==(4-1))
{
    newnode=(struct node*)malloc(sizeof(struct node));
    scanf(" %d",&newnode->data);
    newnode->next=head;
    head=newnode;
    printf("The linked list after insertion at the beginning is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
```

```

        printf("%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
else if(ch==4)
{
    newnode=(struct node*)malloc(sizeof(struct node));
    scanf(" %d",&newnode->data);
    newnode->next=NULL;
    last->next=newnode;
    last=newnode;
    printf("The linked list after deletion at the end is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
        printf("%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
else if(ch==5)
{
    int s;
    scanf("%d",&s);
    newnode=(struct node*)malloc(sizeof(struct node));
    scanf(" %d",&newnode->data);
    if(check(s))
    {
        if(s==head->data)
        {
            newnode->next=head;
            head=newnode;
        }
        else
        {
            ptr=head;
            while(ptr->next->data!=s && ptr->next!=NULL)
            {
                ptr=ptr->next;
            }
            newnode->next=ptr->next;
        }
    }
}

```

```

ptr->next=newnode;
printf("The linked list after insertion before a value is:\n");
ptr=head;
while(ptr!=NULL)
{
    printf("%d",ptr->data);
    ptr=ptr->next;
}
printf("\n");
}
}
else
{
    printf("Value not found in the list\n");
    printf("The linked list after insertion before a value is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
}
else
{
    printf("Value not found in the list\n");
    printf("The linked list after insertion before a value is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
}
else if(ch==6)
{
    int s;
    scanf("%d",&s);
    newnode=(struct node*)malloc(sizeof(struct node));
    scanf(" %d",&newnode->data);

```

```

if(check(s))
{
    last->next=newnode;
    last=newnode;
}
else
{
    ptr=head;
    while(ptr->data!=s)
    {
        ptr=ptr->next;
    }
    newnode->next=ptr->next;
    ptr->next=newnode;
    printf("The linked list after insertion after a value is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
        printf("%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
}
else
{
    printf("Value not found in the list\n");
    printf("The linked list after insertion after a value is:\n");
    ptr=head;
    while(ptr!=NULL)
    {
        printf("%d",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
}

```

Status : Wrong

Marks : 0/1

2. Problem Statement

Bharath is very good at numbers. As he is piled up with many works, he decides to develop programs for a few concepts to simplify his work. As a first step, he tries to arrange even and odd numbers using a linked list. He stores his values in a singly-linked list.

Now he has to write a program such that all the even numbers appear before the odd numbers. Finally, the list is printed in such a way that all even numbers come before odd numbers. Additionally, the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Example

Input:

6

3 1 0 4 30 12

Output:

12 30 4 0 3 1

Explanation:

Even elements: 0 4 30 12

Reversed Even elements: 12 30 4 0

Odd elements: 3 1

So the final list becomes: 12 30 4 0 3 1

Input Format

The first line consists of an integer n representing the size of the linked list.

The second line consists of n integers representing the elements separated by space.

Output Format

The output prints the rearranged list separated by a space.

The list is printed in such a way that all even numbers come before odd numbers

and the even numbers should be in reverse order, while the odd numbers should maintain their original order.

Refer to the sample output for the formatting specifications.

Sample Test Case

Input: 6

3 1 0 4 30 12

Output: 12 30 4 0 3 1

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node
{
    int data;
    struct node *next;
};
struct node*ptr,*newnode,*head=NULL,*last=NULL;
int main()
{
    int a;
    scanf("%d",&a);
    int even[a],odd[a],n=0,m=0;
    for(int i=0;i<a;i++)
    {
        newnode=(struct node*)malloc(sizeof(struct node));
        scanf("%d",&newnode->data);
        newnode->next=NULL;
        if(head==NULL)
        {head=newnode;}
        else
        {last->next=newnode;}
        last=newnode;
    }ptr=head;
    while(ptr!=NULL)
    {if((ptr->data) %2==0)
```



```

{
    even[n++] = ptr->data;
}
else
{
    odd[m++] = ptr->data;
}
ptr = ptr->next;
}

```

```

for(int i = n-1; i >= 0; i--)
{printf("%d ", even[i]);}
for(int i = 0; i < m; i++)
{printf("%d ", odd[i]);}
return 0;
}

```

Status : Wrong

Marks : 0/1

3. Problem Statement

Imagine you are managing the backend of an e-commerce platform. Customers place orders at different times, and the orders are stored in two separate linked lists. The first list holds the orders from morning, and the second list holds the orders from the evening.

Your task is to merge the two lists so that the final list holds all orders in sequence from the morning list followed by the evening orders, in the same order

Input Format

The first line contains an integer n , representing the number of orders in the morning list.

The second line contains n space-separated integers representing the morning orders.

The third line contains an integer m , representing the number of orders in the evening list.

The fourth line contains m space-separated integers representing the evening orders.

Output Format

The output should be a single line containing space-separated integers representing the merged order list, with morning orders followed by evening orders.

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 3

101 102 103

2

104 105

Output: 101 102 103 104 105

Answer

-

Status : Skipped

Marks : 0/1

4. Problem Statement

John is working on evaluating polynomials for his math project. He needs to compute the value of a polynomial at a specific point using a singly linked list representation.

Help John by writing a program that takes a polynomial and a value of x as input, and then outputs the computed value of the polynomial.

Example

Input:

2

13

12
11
1

Output:

36

Explanation:

The degree of the polynomial is 2.

Calculate the value of x_2 : $13 * 12 = 13$.

Calculate the value of x_1 : $12 * 11 = 12$.

Calculate the value of x_0 : $11 * 10 = 11$.

Add the values of x_2 , x_1 and x_0 together: $13 + 12 + 11 = 36$.

Input Format

The first line of input consists of the degree of the polynomial.

The second line consists of the coefficient x_2 .

The third line consists of the coefficient of x_1 .

The fourth line consists of the coefficient x_0 .

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

Output Format

The output is the 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

-

Status : Skipped

Marks : 0/1

5. Problem Statement

Write a program to manage a singly linked list. The program should allow users to perform various operations on the linked list, such as inserting elements at the beginning or end, deleting elements from the beginning or end, inserting before or after a specific value, and deleting elements before or after a specific value. After each operation, the updated linked list should be displayed.

Input Format

The first line contains an integer choice, representing the operation to perform:

- For choice 1 to create the linked list. The next lines contain space-separated integers, with -1 indicating the end of input.
- For choice 2 to display the linked list.
- For choice 3 to insert a node at the beginning. The next line contains an integer data representing the value to insert.
- For choice 4 to insert a node at the end. The next line contains an integer data representing the value to insert.
- For choice 5 to insert a node before a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 6 to insert a node after a specific value. The next line contains two integers: value (existing node value) and data (value to insert).
- For choice 7 to delete a node from the beginning.
- For choice 8 to delete a node from the end.
- For choice 9 to delete a node before a specific value. The next line contains an integer value representing the node before which deletion occurs.
- For choice 10 to delete a node after a specific value. The next line contains an

integer value representing the node after which deletion occurs.

- For choice 11 to exit the program.

Output Format

For choice 1, print "LINKED LIST CREATED".

For choice 2, print the linked list as space-separated integers on a single line. If the list is empty, print "The list is empty".

For choice 3, 4, 5, and 6, print the updated linked list with a message indicating the insertion operation.

For choice 7, 8, 9, and 10, print the updated linked list with a message indicating the deletion operation.

For any operation that is not possible print an appropriate error message such as "Value not found in the list".

For choice 11 terminate the program.

For any invalid option, print "Invalid option! Please try again".

Refer to the sample output for formatting specifications.

Sample Test Case

Input: 1

5

3

7

-1

2

11

Output: LINKED LIST CREATED

5 3 7

Answer

// You are using GCC

Status : Wrong

Marks : 0/1