Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct node{
   int co;
   int exp;
   struct node*next;
};
struct node* createnode(int co,int exp){
   struct node* newnode=(struct node*)malloc(sizeof(struct node));
   newnode->co=co;
   newnode->exp=exp;
   newnode->next=NULL;
   return newnode;
}
void list(struct node **head,int n){
   int co,exp;
   struct node*temp;
   for(int i=0;i<n;i++){
      scanf("%d",&co);
```

```
        scanf("%d",&exp);
        struct node* newnode=createnode(co,exp);
        if(*head==NULL){
            *head=newnode;
            temp=newnode;}
        else{
            temp->next=newnode;
            temp=newnode;
        }
    }
}
int main(){
    int n1,n2;
    struct node *poly1=NULL,*poly2=NULL;
    scanf("%d",&n1);
    list(&poly1,n1);
    scanf("%d",&n2);
    list(&poly2,n2);
    int sum1=0,sum2=0;
    struct node*temp=poly1;
    while(temp!=0){
        sum1+=temp->co;
        temp=temp->next;
    }
    struct node*temp2=poly2;
    while(temp2!=0){
        sum2+=temp2->co;
        temp2=temp2->next;
    }
    printf("%d",sum1+sum2);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 2

Attempt : 1
Total Mark : 10
Marks Obtained : 8.5

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
void insert(int data){

     struct node*newnode=(struct node *)malloc(sizeof(struct node));
     newnode->data=data;
     newnode->next=NULL;
     if(head==NULL){
        head=newnode;
        tail=newnode;
      }
```

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

void display_List(){
    struct node*temp=head;
    while(temp!=NULL){
        printf("%d ",temp->data);
        temp=temp->next;
    }
}
void deleteNode(int pos){
    int count=0;
    struct node*temp=head;
    while(temp!=NULL){
        count+=1;
        temp=temp->next;
    }
    if(pos>count||pos<0){
        printf("Invalid position. Deletion not possible.");

    }
    else{
        int i=1;
        struct node *temp=head;
        struct node* nextnode;
        while(i<pos-1 && temp!=NULL){
            temp=temp->next;
            i++;
        }
        nextnode=temp->next;
        temp->next=nextnode->next;
        free(nextnode);
        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 : Partially correct ***Marks : 8.5/10***

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_COD_Question 3

Attempt : 1
Total Mark : 10
Marks Obtained : 1

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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct node{
   char element;
   struct node* next;
};

struct node*createnode(char element){
   struct node *newnode=(struct node*)malloc(sizeof(struct node));
   newnode->element=element;
   newnode->next=NULL;
```

```
    return newnode;
}
void sll(struct node **head,int n){
    struct node*temp;
    for(int i=0;i<n;i++){
        char c;
        scanf(" %c",&c);
        struct node*newnode=createnode(c);
        if(*head==NULL){
            *head=newnode;
            temp=newnode;
            }
        else{
            temp->next=newnode;
            temp=newnode;
        }
    }
}
void display(struct node *head){
    struct node *temp=head;
    while(temp!=0){
        printf("%c ",temp->element);
        temp=temp->next;
    }
}
void insert(struct node **head,int pos,char c,int n){
    if(pos>=n){
        printf("Invalid index\n");
        printf("Updated list: ");
    }
    else{
        struct node*temp=*head,*newnode;
        int i=0;
        while(i<pos){
            temp=temp->next;
            i++;
        }
        newnode=createnode(c);
        newnode->next=temp->next;
        temp->next=newnode;
        printf("\nUpdated list: ");
    }
```

```
}
int main(){
    int n,pos;
    char c;
    scanf("%d",&n);
    struct node *head=NULL;
    sll(&head,n);
    scanf("%d",&pos);
    scanf("%c",&c);
    insert(&head,pos,c,n);
    display(head);
    return 0;

}
```

Status : Partially correct ***Marks : 1/10***

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_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;
};

void insertAtFront(struct Node**head,int data){
   struct Node*newnode = (struct Node*)malloc(sizeof(struct Node));
   newnode->data=data;
   if(*head==NULL){
      newnode->next=NULL;
      *head=newnode;
   }
   else{
      newnode->next=*head;
      *head=newnode;
   }
}
void printList(struct Node*head){
   struct Node*temp=head;
   while(temp!=NULL){
      printf("%d ",temp->data);
      temp=temp->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: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct node{
   float data;
   struct node*next;
};
struct node* createnode(float data){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->data=data;
   newnode->next=NULL;
   return newnode;
}
void list(struct node**head,int n){
   for(int i=0;i<n;i++){
      float gpa;
      scanf("%f",&gpa);
      struct node *newnode=createnode(gpa);
      newnode->next=*head;
      *head=newnode;
```

```
    }
}
void deletenode(struct node **head,int pos){
    if(*head==NULL)
    return;
    int i=1;
    struct node*temp=*head;
    if(pos==1){
        *head=temp->next;
        free(temp);
        return;
    }
    while(i<pos-1){
        temp=temp->next;
        i++;
    }
    struct node*next=temp->next->next;
    free(temp->next);
    temp->next=next;
}
void display(struct node*head){
    struct node*temp=head;
    while(temp!=NULL){
        printf("GPA: %.1f \n",temp->data);
        temp=temp->next;
    }
}
int main(){
    int n;
    struct node*head=NULL;
    scanf("%d",&n);
    list(&head,n);
    int pos;
    scanf("%d ",&pos);
    deletenode(&head,pos);
    display(head);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_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

```
// You are using GCC
#include <stdio.h>
#include <stdlib.h>
struct node{
   int roll;
   struct node *next;
};

struct node *createnode(int data){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->roll=data;
   newnode->next=NULL;
   return newnode;
}

void list(struct node **head,int n){
   struct node*temp;
   for(int i=0;i<n;i++){
      int roll;
      scanf("%d",&roll);
      struct node*newnode=createnode(roll);
      if(*head==NULL){
         *head=newnode;
         temp=newnode;
      }
      else{
         temp->next=newnode;
         temp=newnode;
```

```
        }
    }
}
void display(struct node*head){
    struct node *temp=head;
    while(temp!=NULL){
        printf("%d ",temp->roll);
        temp=temp->next;
    }
}
int main(){
    int n;
    scanf("%d",&n);
    struct node *head=NULL;
    list(&head,n);
    display(head);
    return 0;

}
```

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_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;
};


// You are using GCC
struct Node * push(struct Node *head,int data){
   struct Node* newnode=(struct Node*)malloc(sizeof(struct Node));
   newnode->data=data;
   if(head==NULL){
      head=newnode;
      newnode->next=NULL;
   }
   else{
      newnode->next=head;
   }
```

```
    return newnode;
}
int printMiddle(struct Node *head){
    int l=0,i=1;
    struct Node*temp=head;
    while(temp!=NULL){
        l++;
        temp=temp->next;
    }
    struct Node*tem=head;
    int mid=l/2;
    while(tem!=NULL && i<=mid){
        tem=tem->next;
        i++;}
    return tem->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: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_week 1_CY

Attempt : 1
Total Mark : 30
Marks Obtained : 20

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>
```

```
typedef struct Term{
   int coeff;
   int exp;
   struct Term* next;
} Term;

Term* createTerm(int coeff, int exp){
   Term* newTerm = (Term*)malloc(sizeof(Term));
   newTerm->coeff = coeff;
   newTerm->exp = exp;
   newTerm->next = NULL;
   return newTerm;
}
Term* insertTerm(Term* head,int coeff,int exp){
   if(coeff == 0) return head;

   Term* newTerm = createTerm(coeff,exp);
   if(!head || head->exp < exp){
      newTerm->next = head;
      return newTerm;
   }
   Term* temp=head;
   Term* prev=NULL;

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

   if (temp && temp->exp == exp) {
      temp->coeff += coeff;
      if (temp->coeff==0){
         if(prev)prev->next=temp->next;
         else head = temp->next;
         free(temp);
      }
      free(newTerm);
   } else {
      newTerm->next = temp;
      if (prev)prev->next=newTerm;
      else head = newTerm;
   }
```

```
    return head;
}
void printPolynomial(Term* head) {
    Term* temp = head;
    while (temp) {
        printf("%dx^%d",temp->coeff,temp->exp);
        if(temp->next) printf(" + ");
        temp=temp->next;
    }
    printf("\n");
}
Term* multiplyPolynomials(Term* poly1, Term* poly2) {
    Term* result= NULL;
    for (Term* i = poly1; i !=NULL; i = i->next){
        for (Term* j=poly2;j !=NULL; j = j->next) {
            int newCoeff = i->coeff * j->coeff;
            int newExp = i->exp + j->exp;
            result = insertTerm(result, newCoeff, newExp);
        }
    }
    return result;
}
Term* readPolynomial(int n){
    Term* head = NULL;
    int coeff,exp;
    for(int i=0;i<n;i++){
        scanf("%d %d", &coeff, &exp);
        head = insertTerm(head,coeff,exp);
    }
    return head;
}
int main(){
    int n,m;
    scanf("%d", &n);
    Term* poly1=readPolynomial(n);
    scanf("%d", &m);
    Term* poly2=readPolynomial(m);

    printPolynomial(poly1);
    printPolynomial(poly2);

    Term* product =multiplePolynomials(poly1, poly2);
    printPolynomial(product);

    return 0;
}
```

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

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>
typedef struct node{
   int coe;
   int expo;
   struct node* next;
}Node;
Node* createNode(int coe,int expo){
   Node* newNode=(Node*)malloc(sizeof(Node));
   newNode->coe=coe;
   newNode->expo=expo;
   newNode->next=NULL;
   return newNode;
}
void insert(Node** head,int coe,int expo){
   Node* newNode=createNode(coe,expo);
   if(expo<0){
      free(newNode);
      return;
   }
   if(*head==NULL||expo>(*head)->expo){
      newNode->next=*head;
      *head=newNode;
      return;
   }
   Node* temp=*head;
```

```
    while(temp->next!=NULL && temp->next->expo>expo){
        temp=temp->next;
    }
    if(temp->next!=NULL && temp->next->expo==expo){
        temp->next->coe=coe;
        free(newNode);
    }
    else{
        newNode->next=temp->next;
        temp->next=newNode;
    }
}
void printList(Node* head){
    if(head==NULL){
        printf("0\n");
        return;
    }
    Node*temp=head;
    int first = 1;
    while(temp!=NULL){
        if(temp->coe!=0){
            if(!first && temp->coe>0){
                printf(" + ");
            }
            if(temp->expo==1){
                printf("%dx",temp->coe);
            }
            else if(temp->expo==0){
                printf("%d",temp->coe);
            }
            else if(temp->expo<0){
                continue;
            }
            else{
                printf("%dx^%d",temp->coe,temp->expo);
            }
            first=0;
        }
        temp=temp->next;
    }
    printf("\n");
}
```

```
void freeList(Node* head){
   while(head!=NULL){
      Node* temp=head;
      head=head->next;
      free(temp);
   }
}
int main(){
   Node* poly1=NULL;
   Node* poly2=NULL;
   int n,coe,expo;
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d %d",&coe,&expo);
      insert(&poly1,coe,expo);
   }
   scanf("%d",&n);
   for(int i=0;i<n;i++){
      scanf("%d %d",&coe,&expo);
      insert(&poly2,coe,expo);
   }
   printList(poly1);
   printList(poly2);
   freeList(poly1);
   freeList(poly2);
}
```

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

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>

#define MAX_TERMS 10

// Structure to represent a polynomial term
typedef struct {
   int coefficient;
   int exponent;
} Term;

// Function to multiply two polynomials
void multiply_polynomials(Term poly1[], int n, Term poly2[], int m, Term result[],
int *result_size) {
   *result_size = 0;
   for (int i = 0; i < n; i++) {
      for (int j = 0; j < m; j++) {
         int coeff = poly1[i].coefficient * poly2[j].coefficient;
         int exp = poly1[i].exponent + poly2[j].exponent;

         // Check if the term with this exponent already exists in the result
         int found = 0;
         for (int k = 0; k < *result_size; k++) {
            if (result[k].exponent == exp) {
               result[k].coefficient += coeff;
               found = 1;
               break;
            }
         }

         // If not found, add a new term
         if (!found) {
            result[*result_size].coefficient = coeff;
            result[*result_size].exponent = exp;
            (*result_size)++;
         }
      }
   }
}
```

```
// Function to print the polynomial
void print_polynomial(Term result[], int result_size) {
    for (int i = 0; i < result_size; i++) {
        if (i > 0 && result[i].coefficient > 0) {
            printf(" + ");
        }

        if (result[i].exponent == 0) {
            printf("%d", result[i].coefficient);
        } else if (result[i].exponent == 1) {
            printf("%dx", result[i].coefficient);
        } else {
            printf("%dx^%d", result[i].coefficient, result[i].exponent);
        }
    }
    printf("\n");
}

// Function to delete a specific exponent from the result polynomial
void delete_term(Term result[], int *result_size, int exp_to_delete) {
    for (int i = 0; i < *result_size; i++) {
        if (result[i].exponent == exp_to_delete) {
            // Shift all terms after the deleted term to the left
            for (int j = i; j < *result_size - 1; j++) {
                result[j] = result[j + 1];
            }
            (*result_size)--; // Reduce the size of the polynomial
            break;
        }
    }
}

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

    // Input the first polynomial
    scanf("%d", &n);
    Term poly1[n];
    for (int i = 0; i < n; i++) {
        scanf("%d %d", &poly1[i].coefficient, &poly1[i].exponent);
    }
    // Input the second polynomial
    scanf("%d", &m);
    Term poly2[m];
    for (int i = 0; i < m; i++) {
        scanf("%d %d", &poly2[i].coefficient, &poly2[i].exponent);
    }

    // Input the exponent to delete
    scanf("%d", &exp_to_delete);

    // Result polynomial after multiplication
    Term result[MAX_TERMS];
    int result_size;
    multiply_polynomials(poly1, n, poly2, m, result, &result_size);

    // Print the result of multiplication
    printf("Result of the multiplication: ");
    print_polynomial(result, result_size);

    // Delete the term with the specified exponent
    delete_term(result, &result_size, exp_to_delete);

    // Print the result after deletion
    printf("Result after deleting the term: ");
    print_polynomial(result, result_size);

    return 0;
}
```

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

Rajalakshmi Engineering College

Name: SARAVANA PERUMAL B
Email: 241901103@rajalakshmi.edu.in
Roll no: 241901103
Phone: 9342922599
Branch: REC
Department: I CSE (CS) FB
Batch: 2028
Degree: B.E - CSE (CS)

Scan to verify results

NeoColab_REC_CS23231_DATA STRUCTURES

REC_DS using C_Week 1_PAH_modified

Attempt : 1
Total Mark : 5
Marks Obtained : 3.4

Section 1 : Coding

1. 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 x2: 13 * 12 = 13.

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

Calculate the value of x0: 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 the degree of the polynomial.

The second line consists of the coefficient x2.

The third line consists of the coefficient of x1.

The fourth line consists of the coefficient x0.

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

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

// Structure to represent each term in the polynomial
typedef struct Term {
   int coefficient;
   int exponent;
   struct Term* next;
} Term;

// Function to create a new term
Term* create_term(int coefficient, int exponent) {
   Term* new_term = (Term*)malloc(sizeof(Term));
   new_term->coefficient = coefficient;
   new_term->exponent = exponent;
   new_term->next = NULL;
   return new_term;
}

// Function to evaluate the polynomial at a given value of x
int evaluate_polynomial(Term* head, int x) {
   int result = 0;
   Term* current = head;

   // Traverse the linked list and compute the value of the polynomial
   while (current != NULL) {
      result += current->coefficient * pow(x, current->exponent);
      current = current->next;
   }

   return result;
}

int main() {
   int degree, coefficient, x_value;

   // Input degree of polynomial
   scanf("%d", &degree);

   // Input coefficients of the polynomial
   Term* head = NULL;
   Term* current = NULL;
   for (int i = degree; i >= 0; i--) {
      scanf("%d", &coefficient);
      Term* new_term = create_term(coefficient, i);

      if (head == NULL) {
          head = new_term;
          current = head;
      } else {
          current->next = new_term;
          current = current->next;
      }
   }

   // Input the value of x at which the polynomial is to be evaluated
   scanf("%d", &x_value);

   // Evaluate the polynomial
   int result = evaluate_polynomial(head, x_value);

   // Output the result
   printf("%d\n", result);

   // Free memory allocated for the linked list
   current = head;
   while (current != NULL) {
      Term* temp = current;
      current = current->next;
      free(temp);
   }

   return 0;
}
```

Status : Correct *Marks : 1/1*

2. 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

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

int main() {
   int n, m;

   // Read the number of morning orders
   scanf("%d", &n);
   int morning_orders[n];

   // Read the morning orders
   for (int i = 0; i < n; i++) {
      scanf("%d", &morning_orders[i]);
   }

   // Read the number of evening orders
   scanf("%d", &m);
   int evening_orders[m];

   // Read the evening orders
   for (int i = 0; i < m; i++) {
      scanf("%d", &evening_orders[i]);
   }

   // Print the merged list (morning orders followed by evening orders)
   for (int i = 0; i < n; i++) {
      printf("%d ", morning_orders[i]);
   }
   for (int i = 0; i < m; i++) {
      printf("%d ", evening_orders[i]);
   }

   printf("\n");

   return 0;
}
```

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

3. 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>

// Define the structure for a linked list node
struct Node {
   int data;
   struct Node* next;
};

// Function to create a new node
struct Node* createNode(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
   newNode->next = NULL;
   return newNode;
}

// Function to push a node at the front of the list (used for even numbers)
void pushFront(struct Node** head, int data) {
   struct Node* newNode = createNode(data);
   newNode->next = *head;
   *head = newNode;
}

// Function to append a node at the end of the list (used for odd numbers)
```

```
void appendEnd(struct Node** head, int data) {
    struct Node* newNode = createNode(data);
    if (*head == NULL) {
        *head = newNode;
        return;
    }
    struct Node* temp = *head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
}

// Function to print the list
void printList(struct Node* head) {
    while (head != NULL) {
        printf("%d ", head->data);
        head = head->next;
    }
}

int main() {
    int n, val;
    scanf("%d", &n);

    struct Node* evenList = NULL;
    struct Node* oddList = NULL;

    for (int i = 0; i < n; i++) {
        scanf("%d", &val);
        if (val % 2 == 0) {
            // Even numbers go to the front of even list (to reverse order)
            pushFront(&evenList, val);
        } else {
            // Odd numbers go to the end of odd list (to keep order)
            appendEnd(&oddList, val);
        }
    }

    // Merge even and odd lists
    struct Node* temp = evenList;
    if (temp == NULL) {
        // If no even numbers, print odd list only
        printList(oddList);
    } else {
        // Traverse to the end of even list
        while (temp->next != NULL)
            temp = temp->next;
        // Link to the odd list
        temp->next = oddList;
        printList(evenList);
    }

    return 0;
}
```

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

4. 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

-

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

// Define the node structure
struct Node {
    int data;
    struct Node* next;
};
```

```
struct Node* head = NULL;

// Function to create the linked list
void createList() {
   int val;
   struct Node* temp = NULL;
   while (1) {
      scanf("%d", &val);
      if (val == -1)
         break;
      struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
      newNode->data = val;
      newNode->next = NULL;
      if (head == NULL) {
         head = newNode;
         temp = head;
      } else {
         temp->next = newNode;
         temp = temp->next;
      }
   }
   printf("LINKED LIST CREATED\n");
}

// Function to display the linked list
void displayList() {
   if (head == NULL) {
      printf("The list is empty\n");
      return;
   }
   struct Node* temp = head;
   while (temp != NULL) {
      printf("%d ", temp->data);
      temp = temp->next;
   }
   printf("\n");
}

// Insert at beginning
void insertAtBeginning(int data) {
   struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
   newNode->data = data;
```

```
    newNode->next = head;
    head = newNode;
    printf("The linked list after insertion at the beginning is:\n");
    displayList();
}

// Insert at end
void insertAtEnd(int data) {
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = NULL;
    if (head == NULL) {
        head = newNode;
    } else {
        struct Node* temp = head;
        while (temp->next != NULL)
            temp = temp->next;
        temp->next = newNode;
    }
    printf("The linked list after insertion at the end is:\n");
    displayList();
}

// Insert before a specific value
void insertBefore(int value, int data) {
    if (head == NULL) {
        printf("Value not found in the list\n");
        return;
    }
    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;

    if (head->data == value) {
        newNode->next = head;
        head = newNode;
        printf("The linked list after insertion before a value is:\n");
        displayList();
        return;
    }

    struct Node* temp = head;
    while (temp->next != NULL && temp->next->data != value)
```

```
        temp = temp->next;

    if (temp->next == NULL) {
        printf("Value not found in the list\n");
        return;
    }

    newNode->next = temp->next;
    temp->next = newNode;
    printf("The linked list after insertion before a value is:\n");
    displayList();
}

// Insert after a specific value
void insertAfter(int value, int data) {
    struct Node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;

    if (temp == NULL) {
        printf("Value not found in the list\n");
        return;
    }

    struct Node* newNode = (struct Node*)malloc(sizeof(struct Node));
    newNode->data = data;
    newNode->next = temp->next;
    temp->next = newNode;
    printf("The linked list after insertion after a value is:\n");
    displayList();
}

// Delete from beginning
void deleteFromBeginning() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }
    struct Node* temp = head;
    head = head->next;
    free(temp);
    printf("The linked list after deletion from the beginning is:\n");
    displayList();
}

// Delete from end
void deleteFromEnd() {
    if (head == NULL) {
        printf("The list is empty\n");
        return;
    }
    if (head->next == NULL) {
        free(head);
        head = NULL;
    } else {
        struct Node* temp = head;
        while (temp->next->next != NULL)
            temp = temp->next;
        free(temp->next);
        temp->next = NULL;
    }
    printf("The linked list after deletion from the end is:\n");
    displayList();
}

// Delete before a specific value
void deleteBefore(int value) {
    if (head == NULL || head->next == NULL || head->data == value) {
        printf("Deletion not possible\n");
        return;
    }

    struct Node* prev = NULL;
    struct Node* curr = head;

    while (curr->next != NULL && curr->next->data != value) {
        prev = curr;
        curr = curr->next;
    }

    if (curr->next == NULL) {
        printf("Value not found in the list\n");
        return;
    }
    if (prev == NULL) {
        head = curr->next;
        free(curr);
    } else {
        prev->next = curr->next;
        free(curr);
    }

    printf("The linked list after deletion before a value is:\n");
    displayList();
}

// Delete after a specific value
void deleteAfter(int value) {
    struct Node* temp = head;
    while (temp != NULL && temp->data != value)
        temp = temp->next;

    if (temp == NULL || temp->next == NULL) {
        printf("Invalid option! Please try again\n");
        return;
    }

    struct Node* toDelete = temp->next;
    temp->next = toDelete->next;
    free(toDelete);
    printf("The linked list after deletion after a value is:\n");
    displayList();
}

// Main function with menu
int main() {
    int choice, data, value;
    while (1) {
        if (scanf("%d", &choice) != 1) break;
        switch (choice) {
            case 1:
                createList();
                break;
            case 2:
                displayList();
                break;
            case 3:
                scanf("%d", &data);
                insertAtBeginning(data);
                break;
            case 4:
                scanf("%d", &data);
                insertAtEnd(data);
                break;
            case 5:
                scanf("%d %d", &value, &data);
                insertBefore(value, data);
                break;
            case 6:
                scanf("%d %d", &value, &data);
                insertAfter(value, data);
                break;
            case 7:
                deleteFromBeginning();
                break;
            case 8:
                deleteFromEnd();
                break;
            case 9:
                scanf("%d", &value);
                deleteBefore(value);
                break;
            case 10:
                scanf("%d", &value);
                deleteAfter(value);
                break;
            case 11:
                return 0;
            default:
                printf("Invalid option! Please try again\n");
        }
    }
    return 0;
}
```

Status : Partially correct ***Marks : 0.4/1***