Rajalakshmi Engineering College

Name: Zaina Raheen A.P
Email: 241801328@rajalakshmi.edu.in
Roll no: 241801328
Phone: 9384899474
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 1

Attempt : 2
Total Mark : 10
Marks Obtained : 10

Section 1 : Coding

1. Problem Statement

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

The polynomials will be represented as a linked list, where each node of the linked list contains a coefficient and an exponent. The polynomial is represented in the standard form with descending order of exponents.

Input Format

The first line of input consists of an integer n, representing the number of terms in the first polynomial.

The following n lines of input consist of two integers each: the coefficient and the exponent of the term in the first polynomial.

The next line of input consists of an integer m, representing the number of terms in the second polynomial.

The following m lines of input consist of two integers each: the coefficient and the exponent of the term in the second polynomial.

Output Format

The output prints the sum of the coefficients of the polynomials.

Sample Test Case

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

Output: 18

Answer

```
// You are using GCC
#include<stdio.h>
#include<stdlib.h>
struct node{
    int coef;
    int exp;
    struct node*next;
};
struct node*createnode(int coef,int exp){
   struct node*newnode=(struct node*)malloc(sizeof(struct node));
   newnode->coef=coef;
   newnode->exp=exp;
   newnode->next=NULL;
   return newnode;
}
void insertnode(struct node** head,int coef,int exp){
    struct node*newnode=createnode(coef,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;
    struct node*p1=poly1, *p2=poly2;
    while(p1!=NULL || p2!=NULL){
        if(p1!=NULL && (p2==NULL || p1->exp > p2->exp)){
            insertnode(&result,p1->coef,p1->exp);
            p1=p1->next;
        }else if(p2!=NULL && (p1==NULL || p2->exp > p1->exp)){
            insertnode(&result,p2->coef,p2->exp);
            p2=p2->next;
        }else{
            int sumcoef=p1->coef+p2->coef;
            if(sumcoef!=0){
                insertnode(&result,sumcoef,p1->exp);
            }
            p1=p1->next;
            p2=p2->next;
        }
    }
    return result;
}
int sumcoefficients(struct node*head){
    int sum=0;
    struct node*temp=head;
    while(temp!=NULL){
        sum+=temp->coef;
        temp=temp->next;
    }
    return sum;
}
void freepolynomial(struct node*head){
    struct node*temp;
    while(head!=NULL){
        temp=head;
```

```
        head=head->next;
        free(temp);
    }
}
int main(){
    int n,m;
    struct node*poly1=NULL,*poly2=NULL;
    scanf("%d", &n);
    for(int i=0;i<n;i++){
        int coef,exp;
        scanf("%d %d", &coef, &exp);
        insertnode(&poly1, coef, exp);
    }
    scanf("%d", &m);
    for(int i=0;i<m;i++){
        int coef,exp;
        scanf("%d %d", &coef, &exp);
        insertnode(&poly2, coef, exp);
    }
    struct node*result=addpolynomials(poly1, poly2);
    printf("%d\n",sumcoefficients(result));
    freepolynomial(poly1);
    freepolynomial(poly2);
    freepolynomial(result);
    return 0;
}
```

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

Rajalakshmi Engineering College

Name: Zaina Raheen A.P
Email: 241801328@rajalakshmi.edu.in
Roll no: 241801328
Phone: 9384899474
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 : 2
Total Mark : 30
Marks Obtained : 22.5

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

```
// 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 insertnode(node** head ,int coeff,int exp) {
   node* newnode=createnode(coeff,exp);
   if(*head==NULL){
      *head=newnode;
   }else{
      node* temp=*head;
      while(temp->next!=NULL){
         temp=temp->next;
      }
      temp->next=newnode;
   }
}
void printpolynomial(node* head){
   node* temp=head;
   int ft=1;
   while(temp!=NULL){
      if(temp->coeff!=0){
         if(!ft){
            printf("+");
         }else{
            ft=0;
         }
         if(temp->exp==0){
            printf("%d",temp->coeff);
         }else if(temp->exp==1){
            printf("%dx",temp->coeff);
         }else{
            printf("%dx^%d",temp->coeff,temp->exp);
         }
```

```
        }
        temp=temp->next;
    }
    printf("\n");
}
node* multiplypolynomials(node* poly1,node* poly2){
    node* res=NULL;
    node* temp1=poly1;
    node* temp2=poly2;
    while(temp1!=NULL){
        temp2=poly2;
        while(temp2!=NULL){
            int coeff=temp1->coeff*temp2->coeff;
            int exp=temp1->exp+temp2->exp;
            node* current=res;
            node* prev=NULL;
            int found=0;
            while(current!=NULL){
                if(current->exp==exp){
                    current->coeff+=coeff;
                    found=1;
                    break;
                }
                prev=current;
                current=current->next;
            }
            if(!found){
                node* newnode=createnode(coeff,exp);
                if(res==NULL){
                    res=newnode;
                }else if(res->exp < exp){
                    newnode->next=res;
                    res=newnode;
                }else{
                    current=res;
                    prev=NULL;
                    while(current!=NULL && current->exp > exp){
                        prev=current;
                        current=current->next;
                    }
                    if(prev==NULL){
                        newnode->next=res;
```

```
                res=newnode;
            }else{
                newnode->next=current;
                prev->next=newnode;
            }
        }
    }
    temp2=temp2->next;
  }
  temp1=temp1->next;
 }
 return res;
}
void freeList(node* head){
 node* temp;
 while(head!=NULL){
  temp=head;
  head=head->next;
  free(temp);
 }
}
int main(){
 int n,m,coeff,exp;
 node* poly1=NULL;
 node* poly2=NULL;
 scanf("%d",&n);
 for(int i=0;i<n;i++){
  scanf("%d %d",&coeff,&exp);
  insertnode(&poly1,coeff,exp);
 }
 scanf("%d",&m);
 for(int i=0;i<m;i++){
  scanf("%d %d",&coeff,&exp);
  insertnode(&poly2,coeff,exp);
 }
 printpolynomial(poly1);
 printpolynomial(poly2);
 node* res=multiplypolynomials(poly1,poly2);
 printpolynomial(res);
 freeList(poly1);
 freeList(poly2);
 freeList(res);
```

}

Status : Partially correct ***Marks : 5/10***

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 co,e;
    struct node* next;
};
void insert(struct node** head,int co,int e){
    struct node* newnode=(struct node*)malloc(sizeof(struct node));
    newnode->co=co;
    newnode->e=e;
    newnode->next=NULL;

    if(!*head || (*head)->e>e){
        newnode->next=*head;
        *head=newnode;
        return;
    }
    struct node* current=*head;
    while(current->next && current->next->e<e)
    current=current->next;

    newnode->next=current->next;
    current->next=newnode;
}
void readpolynomial(struct node**head){
    int co,e;
    while(scanf("%d %d",&co,&e),co||e)insert(head,co,e);
}

struct node* addpolynomial(struct node*p1,struct node*p2){
    struct node*result=NULL,*temp;
    while(p1&& p2){
        if(p1->e==p2->e){
            int sc=p1->co+p2->co;
            if(sc)
            insert(&result,sc,p1->e);
            p1=p1->next;
            p2=p2->next;
        }
        else if(p1->e<p2->e){
            insert(&result,p1->co,p1->e);
```

```
            p1=p1->next;
        }else{
            insert(&result,p2->co,p2->e);
            p2=p2->next;
        }
    }
    while(p1){
        insert(&result,p1->co,p1->e);
        p1=p1->next;
    }
    while(p2){
        insert(&result,p2->co,p2->e);
        p2=p2->next;
    }
    return result;
}
void printpolynomial(struct node*head){
    if(!head){
        printf("0\n");
        return;
    }
    while(head){
        printf("%dx^%d",head->co,head->e);
        if(head->next)
        printf("+");
        head=head->next;
    }
    printf("\n");

}
int main()
{
    struct node*p1=NULL,*p2=NULL;
    readpolynomial(&p1);
    readpolynomial(&p2);

    printpolynomial(p1);
    printpolynomial(p2);

    struct node*result=addpolynomial(p1,p2);
    printpolynomial(result);
    return 0;
```

}

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>

typedef struct Poly{
    int coefficient;
    int exponential;
    struct Poly* next;
}Node;

Node* newNode(int coefficient,int exponential){
    Node* polynomial=(Node*)malloc(sizeof(Node));
    polynomial->coefficient=coefficient;
    polynomial->exponential=exponential;
    polynomial->next=NULL;
    return polynomial;
}

void insertNode(Node** head,int coefficient,int exponential){
    Node* newnode=newNode(coefficient,exponential);
```

```
    if(*head==NULL){
        *head=newnode;
        return;
    }
    Node* temp=*head;
    Node* prev=NULL;

    while(temp!=NULL && temp->exponential >exponential){
        prev=temp;
        temp=temp->next;
    }
    if(temp!=NULL && temp->exponential==exponential){
        temp->coefficient+=coefficient;
        free(newnode);
        return;
    }
    if(prev==NULL){
        newnode->next=*head;
        *head=newnode;
    }else{
        newnode->next=temp;
        prev->next=newnode;
    }
}

void print(Node* head){
    if(head==NULL){
        printf("0\n");
        return;
    }
    Node* temp=head;
    int ft=1;

    while(temp!=NULL){
        if(ft){
            ft=0;
        }else{
            printf(" + ");
        }

        if(temp->exponential==0){
            printf("%d",temp->coefficient);
```

```
        }
        else if (temp->exponential==1){
            printf("%dx",temp->coefficient);
        }
        else{
            printf("%dx^%d  ",temp->coefficient,temp->exponential);
        }

        temp=temp->next;
    }
    printf("\n");
}

Node* multiply(Node* poly1,Node* poly2){
    Node* result=NULL;
    Node* temp1=poly1;

    while(temp1!=NULL){
        Node* temp2=poly2;
        while(temp2!=NULL){
            int coeff=temp1->coefficient*temp2->coefficient;
            int exp=temp1->exponential + temp2->exponential;
            insertNode(&result,coeff,exp);
            temp2=temp2->next;
        }
        temp1=temp1->next;
    }
    return result;
}
Node* deleteTerm(Node* head,int exponential){
    Node* current=head;
    Node* prev=NULL;

    while(current!=NULL && current->exponential !=exponential){
        prev=current;
        current=current->next;
    }
    if(current==NULL){
        return head;
    }
    if(prev==NULL){
        head=current->next;
```

```
    }else{
        prev->next=current->next;
    }
    free(current);
    return head;
}
int main(){
    Node* poly1=NULL;
    Node* poly2=NULL;
    int n,coefficient,exponential,input;
    scanf("%d", &n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coefficient,&exponential);
        insertNode(&poly1,coefficient,exponential);
    }
    scanf("%d", &n);
    for(int i=0;i<n;i++){
        scanf("%d %d",&coefficient,&exponential);
        insertNode(&poly2,coefficient,exponential);
    }

    scanf("%d", &input);
    Node* result=multiply(poly1,poly2);
    printf("Result of the multiplication: ");
    print(result);

    Node* resultAfterDelete=deleteTerm(result, input);
    printf("Result after deleting the term: ");
    print(resultAfterDelete);

    return 0;
}
```

Status : Partially correct ***Marks : 7.5/10***

Rajalakshmi Engineering College

Name: Zaina Raheen A.P
Email: 241801328@rajalakshmi.edu.in
Roll no: 241801328
Phone: 9384899474
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 : 3.35

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

Node* head=NULL;

void insertAB(int data){
    Node* newNode=(Node*)malloc(sizeof(Node));
    if(newNode==NULL){
        printf("Memory allocation failed\n");
        return;
    }
    newNode->data=data;
    newNode->next=head;
    head=newNode;
}

void insertAE(int data){
    Node* newNode=(Node*)malloc(sizeof(Node));
    newNode->data=data;
    newNode->next=NULL;

    if(head==NULL){
        head=newNode;
        return;
```

```
    }
    Node* current=head;
    while(current->next!=NULL){
        current=current->next;
    }
    current->next=newNode;
}

void deleteFB(){
    if(head==NULL)return;
    Node* temp=head;
    head=head->next;
    free(temp);
}

void deleteFE(){
    if(head==NULL) return;

    if(head->next==NULL){
        free(head);
        head=NULL;
        return;
    }
    Node* current=head;
    while(current->next->next!=NULL){
        current=current->next;
    }
    free(current->next);
    current->next=NULL;
}

void insertB(int value,int newValue){
    if(head==NULL){
        printf("Value not found in the list\n");
        return;
    }
    if(head->data==value){
        insertAB(newValue);
        return;
    }
    Node* current=head->next;
    Node* previous=head;
```

```
    while(current!=NULL){
        if(current->data==value){
            Node* newNode=(Node*)malloc(sizeof(Node));
            newNode->data=newValue;
            newNode->next=current;
            previous->next=newNode;
            return;
        }
        previous=current;
        current=current->next;
    }
    printf("Value not found in the list\n");
}
void insertA(int value,int newValue){
    Node* current=head;
    while(current!=NULL){
        if(current->data==value){
            Node* newNode=(Node*)malloc(sizeof(Node));
            newNode->data=newValue;
            newNode->next=current->next;
            current->next=newNode;
            return;
        }
        current=current->next;
    }
    printf("Value not found in the list\n");
}

void deleteB(int value){
    if(head==NULL || head->next==NULL){
        printf("Value not found in the lisy\n");
        return;
    }
    if(head->next->data==value){
        deleteFB();
        return;
    }
    Node* current=head->next->next;
    Node* previous=head->next;
    Node* beforePrevious=head;
    while(current!=NULL){
        if(current->data==value){
```

```
            beforePrevious->next=current;
            free(previous);
            return;
        }
        beforePrevious=previous;
        previous=current;
        current=current->next;
    }
    printf("Value not found in the list\n");
}
void deleteA(int value){
    Node* current=head;
    while(current!=NULL && current->next!=NULL){
        if(current->data==value){
            Node* temp=current->next;
            current->next=temp->next;
            free(temp);
            return;
        }
        current=current->next;
    }
    printf("Value not found in the list\n");
}
void displayList(){
    if(head==NULL){
        printf("The list is empty\n");
        return;
    }
    Node* current=head;
    while(current!=NULL){
        printf("%d ",current->data);
        current=current->next;
    }
    printf("\n");
}
int main(){
    int choice,data,value,newValue;
    while(1){
        scanf("%d",&choice);
        switch(choice){
            case 1:
                while(1){
```

```
            scanf("%d",&data);
            if(data==-1)break;
            insertAE(data);
        }
        printf("LINKED LIST CREATED\n");
        break;
    case 2:
        displayList();
        break;
    case 3:
        scanf("%d",&data);
        insertAB(data);
        printf("The linked list after insertion at the beginning is:\n");
        displayList();
        break;
    case 4:
        scanf("%d",&data);
        insertAE(data);
        printf("The linked list after insertion at the end is:\n");
        displayList();
        break;
    case 5:
        scanf("%d %d",&value,&newValue);
        insertB(value,newValue);
        printf("The linked list after insertion before a value is:\n");
        displayList();
        break;
    case 6:
        scanf("%d %d",&value,&newValue);
        insertA(value,newValue);
        printf("The linked list after insertion after a value is:\n");
        displayList();
        break;
    case 7:
        deleteFB();
        printf("The linked list after deletion from the beginning is:\n");
        displayList();
        break;
    case 8:
        scanf("%d",&value);
        deleteB(value);
        printf("The linked list after deletion before a value is:\n");
```

```
            displayList();
            break;
        case 9:
            scanf("%d",&value);
            deleteA(value);
            printf("The linked list after deletion after a value is:\n");
            displayList();
            break;
        case 10:
            deleteFE();
            printf("The linked list after deletion from the end is:\n");
            displayList();
            break;
        case 11:
            return 0;
        default:
            printf("Invalid option! Please try again\n");
    }
  }
  return 0;
}
```

Status : Partially correct **Marks : 0.35/1**

2. 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.

Rajalakshmi Engineering College

Name: Zaina Raheen A.P
Email: 241801328@rajalakshmi.edu.in
Roll no: 241801328
Phone: 9384899474
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. 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***

2. 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***

3. 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***

4. 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***

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

Answer

Binary search

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

6. 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***

7. 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***

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. 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***

10. 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***