KSHIPRA MALVIYA

DSA Assignment

Q1

#include<stdio.h>

#include<stdlib.h>

struct node

{

    int roll;

    char name[50];

    int marks;

    struct node\* next;

};

int main()

{

    int n;

    printf("Enter the number of students : ");

    scanf("%d",&n);

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    printf("Enter details (roll no. , name , marks) of student 1\n");

    scanf("%d",&head->roll);

    scanf("%s",&head->name);

    scanf("%d",&head->marks);

    fflush(stdin);

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<n; x++)

    {

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        printf("Enter details (roll no. , name , marks) of student %d\n",x+1);

        scanf("%d",&llnode->roll);

        scanf("%s",&llnode->name);

        scanf("%d",&llnode->marks);

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    printf("\nLinked List\n\n");

    temp=head;

    while(temp!=NULL)

    {

        printf("(%d , %s , %d)  ->  ", temp->roll, temp->name, temp->marks);

        temp=temp->next;

    }

    printf("NULL\n");

    return 0;

}

Q2

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

int main()

{

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    printf("Enter 10 elements in the linked list\n");

    int n;

    scanf("%d",&n);

    head->data=n;

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<=9; x++)

    {

        scanf("%d",&n);

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        llnode->data=n;

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    printf("\nLinked List\n\n");

    struct node\* t=head;

    while(t!=NULL)

    {

        printf("%d -> ",t->data);

        t=t->next;

    }

    printf("NULL\n");

    return 0;

}

Q3

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

int main()

{

    int size;

    printf("Enter size of the array : ");

    scanf("%d",&size);

    int ar[size];

    printf("Enter elements\n");

    for(int x=0; x<size; x++)

    {

        scanf("%d",&ar[x]);

    }

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    head->data=ar[0];

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<size; x++)

    {

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        llnode->data=ar[x];

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    printf("\nLinked List\n\n");

    struct node\* t=head;

    while(t!=NULL)

    {

        printf("%d -> ",t->data);

        t=t->next;

    }

    printf("NULL\n");

    return 0;

}

Q4

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

struct node\* createLL()

{

    int size;

    printf("Enter size of the linked list : ");

    scanf("%d",&size);

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    printf("Enter elements in the linked list\n");

    int n;

    scanf("%d",&n);

    head->data=n;

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<size; x++)

    {

        scanf("%d",&n);

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        llnode->data=n;

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    return head;

}

int size(struct node\* head)

{

    int count=0;

    struct node\* temp=head;

    while(temp!=NULL)

    {

        count++;

        temp=temp->next;

    }

    return count;

}

int equal(struct node\* head1, struct node\* head2)

{

    int size1=size(head1);

    int size2=size(head2);

    if(size1!=size2)

    {

        return 0;

    }

    else

    {

        struct node\* t1=head1;

        struct node\* t2=head2;

        while(t1!=NULL)

        {

            if(t1->data!=t2->data)

            {

                return 0;

            }

            t1=t1->next;

            t2=t2->next;

        }

    }

    return 1;

}

void printLL(struct node\* head)

{

    struct node\* temp=head;

    while(temp!=NULL)

    {

        printf("%d -> ",temp->data);

        temp=temp->next;

    }

    printf("NULL\n");

}

int main()

{

    printf("For 1st Linked List\n");

    struct node\* head1=createLL();

    printf("For 2nd Linked List\n");

    struct node\* head2=createLL();

    printf("\n\n1st Linked List\n");

    printLL(head1);

    printf("\n2nd Linked List\n");

    printLL(head2);

    if(equal(head1,head2)==1)

    {

        printf("\n\nEqual Linked List\n");

    }

    else

    {

        printf("\n\nUnequal Linked List\n");

    }

    return 0;

}

Q5

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

int size(struct node\* head)

{

    int count=0;

    struct node\* temp=head;

    while(temp!=NULL)

    {

        count++;

        temp=temp->next;

    }

    return count;

}

struct node\* findIntersectionPoint(struct node\* head1, struct node\* head2)

{

    int size1=size(head1);

    int size2=size(head2);

    struct node\* t1=(struct node\*)malloc(sizeof(struct node));

    struct node\* t2=(struct node\*)malloc(sizeof(struct node));

    t1=head1;

    t2=head2;

    if(size1>size2)

    {

        int diff=size1-size2;

        while(diff--)

        {

            t1=t1->next;

        }

    }

    else

    {

        int diff=size2-size1;

        while(diff--)

        {

            t2=t2->next;

        }

    }

    while(t1!=NULL)

    {

        if(t1==t2)

        {

            return t1;

        }

        t1=t1->next;

        t2=t2->next;

    }

    return NULL;

}

int main()

{

    struct node\* n1=(struct node\*)malloc(sizeof(struct node));

    struct node\* n2=(struct node\*)malloc(sizeof(struct node));

    struct node\* n3=(struct node\*)malloc(sizeof(struct node));

    struct node\* n4=(struct node\*)malloc(sizeof(struct node));

    struct node\* n5=(struct node\*)malloc(sizeof(struct node));

    struct node\* n6=(struct node\*)malloc(sizeof(struct node));

    struct node\* n7=(struct node\*)malloc(sizeof(struct node));

    struct node\* n8=(struct node\*)malloc(sizeof(struct node));

    n1->data=1; n1->next=n2;

    n2->data=2; n2->next=n3;

    n3->data=3; n3->next=n4;

    n4->data=4; n4->next=n5;

    n5->data=5; n5->next=n6;

    n6->data=6; n6->next=NULL;

    n7->data=9; n7->next=n8;

    n8->data=10; n8->next=n5;

    struct node\* intersection=findIntersectionPoint(n1,n7);

    if(intersection!=NULL)

    {

        printf("Intersects at the node with data %d\n",intersection->data);

    }

    else

    {

        printf("The linked lists don't intersect\n");

    }

    return 0;

}

Q6

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

struct node\* createLL()

{

    int size;

    printf("Enter size of the linked list : ");

    scanf("%d",&size);

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    printf("Enter elements in the linked list\n");

    int n;

    scanf("%d",&n);

    head->data=n;

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<size; x++)

    {

        scanf("%d",&n);

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        llnode->data=n;

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    return head;

}

struct node\* reverse(struct node\* head)

{

    struct node\* prev=NULL;

    struct node\* curr=head;

    while(curr!=NULL)

    {

        struct node\* nex=curr->next;

        curr->next=prev;

        prev=curr;

        curr=nex;

    }

    return prev;

}

void printLL(struct node\* head)

{

    struct node\* temp=head;

    while(temp!=NULL)

    {

        printf("%d -> ",temp->data);

        temp=temp->next;

    }

    printf("NULL\n");

}

int main()

{

    struct node\* head=createLL();

    printf("\n\nOriginal Linked List\n");

    printLL(head);

    printf("\nReversed Linked List\n");

    printLL(reverse(head));

    return 0;

}

Q7

#include<stdlib.h>

#include<stdio.h>

struct node

{

    int data;

    struct node\* next;

};

struct node\* createLL()

{

    int size;

    printf("Enter size of the linked list : ");

    scanf("%d",&size);

    struct node\* head=(struct node\*)malloc(sizeof(struct node));

    printf("Enter elements in the linked list\n");

    int n;

    scanf("%d",&n);

    head->data=n;

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    temp=head;

    for(int x=1; x<size; x++)

    {

        scanf("%d",&n);

        struct node\* llnode=(struct node\*)malloc(sizeof(struct node));

        llnode->data=n;

        temp->next=llnode;

        temp=llnode;

    }

    temp->next=NULL;

    return head;

}

int size(struct node\* head)

{

    int count=0;

    struct node\* temp=head;

    while(temp!=NULL)

    {

        count++;

        temp=temp->next;

    }

    return count;

}

struct node\* merge(struct node\* head1, struct node\* head2)

{

    struct node\* t1=head1;

    struct node\* t2=head2;

    struct node\* newhead=(struct node\*)malloc(sizeof(struct node));

    struct node\* temp=(struct node\*)malloc(sizeof(struct node));

    if(t1->data < t2->data)

    {

        newhead=t1;

        t1=t1->next;

    }

    else

    {

        newhead=t2;

        t2=t2->next;

    }

    temp=newhead;

    while(t1!=NULL && t2!=NULL)

    {

        if(t1->data < t2->data)

        {

            temp->next=t1;

            temp=t1;

            t1=t1->next;

        }

        else

        {

            temp->next=t2;

            temp=t2;

            t2=t2->next;

        }

    }

    if(t1!=NULL)

    {

        temp->next=t1;

    }

    if(t2!=NULL)

    {

        temp->next=t2;

    }

    return newhead;

}

void printLL(struct node\* head)

{

    struct node\* temp=head;

    while(temp!=NULL)

    {

        printf("%d -> ",temp->data);

        temp=temp->next;

    }

    printf("NULL\n");

}

int main()

{

    printf("For 1st Linked List\n");

    struct node\* head1=createLL();

    printf("For 2nd Linked List\n");

    struct node\* head2=createLL();

    printf("\n\n1st Linked List\n");

    printLL(head1);

    printf("\n2nd Linked List\n");

    printLL(head2);

    printf("\n\nMerged Linked List\n");

    printLL(merge(head1,head2));

    return 0;

}

Q8

#include <stdio.h>

#include<stdlib.h>

struct node

{

    int c;

    int e;

    struct node \*next;

};

struct node \*createLL()

{

    int size;

    printf("Enter the number of terms in the polynomial : ");

    scanf("%d", &size);

    struct node \*head = (struct node \*)malloc(sizeof(struct node));

    printf("Enter element 1 (coefficient and power of x)\n");

    int coeff, exp;

    scanf("%d%d", &coeff, &exp);

    head->c = coeff;

    head->e = exp;

    struct node \*temp = (struct node \*)malloc(sizeof(struct node));

    temp = head;

    for (int x = 1; x < size; x++)

    {

        printf("Enter element %d (coefficient and power of x)\n", x + 1);

        scanf("%d%d", &coeff, &exp);

        struct node \*llnode = (struct node \*)malloc(sizeof(struct node));

        llnode->c = coeff;

        llnode->e = exp;

        temp->next = llnode;

        temp = llnode;

    }

    temp->next = NULL;

    return head;

}

struct node\* insert(struct node\* head, int coeff, int exp)

{

    struct node\* temp;

    struct node\* newPoly=(struct node \*)malloc(sizeof(struct node));

    newPoly->c=coeff;

    newPoly->e=exp;

    newPoly->next=NULL;

    if(head==NULL || exp > head->e)

    {

        newPoly->next=head;

        head=newPoly;

    }

    else

    {

        temp=head;

        while(temp->next!=NULL && temp->next->e >= exp)

        {

            temp=temp->next;

        }

        newPoly->next=temp->next;

        temp->next=newPoly;

    }

    return head;

}

struct node \*product(struct node \*head1, struct node \*head2)

{

    struct node \*newhead = (struct node \*)malloc(sizeof(struct node));

    struct node \*t1 = head1;

    struct node \*temp = newhead;

    while (t1 != NULL)

    {

        struct node \*t2 = head2;

        while (t2 != NULL)

        {

            int coeff = (t1->c) \* (t2->c);

            int exp = (t1->e) + (t2->e);

            newhead=insert(newhead,coeff,exp);

            t2=t2->next;

        }

        t1=t1->next;

    }

    temp=newhead;

    while(temp->next!=NULL)

    {

        if(temp->e==temp->next->e)

        {

            temp->c=temp->c + temp->next->c;

            struct node\* tt=temp->next;

            temp->next=temp->next->next;

            free(tt);

        }

        else

        {

            temp=temp->next;

        }

    }

    return newhead;

}

void printLL(struct node\* head)

{

    struct node\* temp=head;

    while(temp->next!=NULL)

    {

        printf("%dx^%d + ",temp->c,temp->e);

        temp=temp->next;

    }

    printf("%dx^%d",temp->c,temp->e);

}

int main()

{

    printf("For the first polynomial\n");

    struct node\* head1=createLL();

    printf("For the second polynomial\n");

    struct node\* head2=createLL();

    printf("\np1    : ");

    printLL(head1);

    printf("\np2    : ");

    printLL(head2);

    printf("\np1.p2 : ");

    printLL(product(head1,head2));

    return 0;

}