Name: Davasam Karthikeya

Reg No: 230962326

Class and Lab Batch: AIMLB and B2

Date: 04/01/2025

Question 1)

#include <stdio.h>

#include <stdlib.h>

typedef struct Node{

    int data;

    struct Node \*next;

    struct Node \*prev;

}\*node\_ptr;

node\_ptr create\_node(int data){

    node\_ptr ptr = (node\_ptr) malloc(sizeof(node\_ptr));

    ptr->data = data;

    ptr->next = NULL;

    ptr->prev = NULL;

}

void display(node\_ptr head){

    node\_ptr ptr;

    ptr = head;

    while(ptr->next != NULL){

        printf("%d -> ",ptr->data);

        ptr = ptr->next;

    }

    printf("%d \n",ptr->data);

}

node\_ptr insert\_end(node\_ptr head,int item){

    node\_ptr new\_node = create\_node(item);

    if(head == NULL)return new\_node;

    node\_ptr ptr;

    ptr = head;

    while(ptr->next != NULL)ptr = ptr->next;

    new\_node->prev = ptr;

    ptr->next = new\_node;

    return head;

}

node\_ptr insert\_beg(node\_ptr head,int item){

    node\_ptr new\_node = create\_node(item);

    if(head == NULL)return new\_node;

    new\_node->next = head;

    head->prev = new\_node;

    return new\_node;

}

node\_ptr insert\_left(node\_ptr head, int key, int item){

    node\_ptr new\_node = create\_node(item);

    if(head == NULL)return NULL;

    node\_ptr ptr,ptr1;

    ptr = NULL;ptr1 = head;

    while(ptr1 != NULL){

        if(ptr1->data == key){

            if(ptr == NULL){

                ptr1->prev = new\_node;

                new\_node->next = ptr1;

                return new\_node;

            }

            ptr->next = new\_node;

            new\_node->prev = ptr;

            new\_node->next = ptr1;

            ptr1->prev = new\_node;

            return head;

        }

        ptr = ptr1;

        ptr1 = ptr1->next;

    }

}

void search\_item(node\_ptr head, int item){

    node\_ptr ptr;

    int count = 0;

    ptr = head;

    while(ptr != NULL){

        if(ptr->data == item)count++;

        ptr = ptr->next;

    }

    printf("Found Element %d at %d Instances!\n",item,count);

}

void delete\_all(node\_ptr head, int item){

    node\_ptr ptr,ptr\_prev;

    ptr\_prev = NULL;ptr = head;

    while(ptr != NULL){

        if(ptr->data == item){

            if(ptr != NULL){

                ptr\_prev->next = ptr->next;

                if(ptr->next != NULL)ptr->next->prev = ptr\_prev;

            }

            free(ptr);

            ptr = ptr\_prev;

        }

        ptr\_prev = ptr;

        ptr = ptr->next;

    }

}

int main(){

    node\_ptr head;

    head = NULL;

    head = insert\_beg(head,2);

    head = insert\_beg(head,3);

    head = insert\_beg(head,4);

    head = insert\_beg(head,1);

    display(head);

    head = insert\_left(head,3,8);

    display(head);

    search\_item(head,8);

    delete\_all(head, 8);

    display(head);

    return 0;

}

Output:

A screen shot of a computer

Description automatically generated

Question 2)

#include <stdio.h>

#include <stdlib.h>

typedef struct Node{

    int data;

    struct Node \*left;

    struct Node \*right;

}\*node\_ptr;

node\_ptr create\_node(int data){

    node\_ptr ptr = (node\_ptr) malloc(sizeof(node\_ptr));

    ptr->data = data;

    ptr->left = NULL;

    ptr->right = NULL;

}

node\_ptr insert\_BT(node\_ptr head, int item){

    node\_ptr new\_node = create\_node(item);

    if(head == NULL)return new\_node;

    node\_ptr ptr,ptr\_prev;

    ptr = head; ptr\_prev = NULL;

    while(ptr != NULL){

        ptr\_prev = ptr;

        if(ptr->data > item)ptr = ptr->left;

        else if(ptr->data < item) ptr = ptr->right;

        else{

            printf("Key %d already Exists! \n",item);

            return head;

        }

    }

    if(ptr\_prev->data > item)ptr\_prev->left = new\_node;

    else ptr\_prev->right = new\_node;

    return head;

}

void recInOrder(node\_ptr ptr){

    if(ptr != NULL){

        recInOrder(ptr->left);

        printf("%d ",ptr->data);

        recInOrder(ptr->right);

    }

}

void recPreOrder(node\_ptr ptr){

    if(ptr != NULL){

        printf("%d ",ptr->data);

        recPreOrder(ptr->left);

        recPreOrder(ptr->right);

    }

}

void recPostOrder(node\_ptr ptr){

    if(ptr != NULL){

        recPostOrder(ptr->left);

        recPostOrder(ptr->right);

        printf("%d ",ptr->data);

    }

}

int main(){

    node\_ptr root;

    root = NULL;

    root = insert\_BT(root,2);

    root = insert\_BT(root,3);

    root = insert\_BT(root,4);

    root = insert\_BT(root,4);

    printf("RecInOrder: ");recInOrder(root);

    printf("\nRecPreOrder: ");recPreOrder(root);

    printf("\nRecPostOrder: ");recPostOrder(root);printf("\n");

    return 0;

}

Output:

A black background with white text

Description automatically generated

Question 3)

#include <stdio.h>

#include <stdlib.h>

typedef struct Node{

    int data;

    struct Node \*next;

}\*node\_ptr;

typedef struct NodeArray{

    int data;

    struct NodeArray \*next;

    struct Node \*right;

}\*nodearr\_ptr;

void GenerateGraph(){

    int nov;

    printf("Enter total no of vertices: ");scanf("%d",&nov);

    nodearr\_ptr AdjacencyList;

    AdjacencyList = NULL;

    for(int i = 0;i<nov;i++){

        nodearr\_ptr new\_node = (nodearr\_ptr) malloc(sizeof(nodearr\_ptr));

        new\_node->data = i;

        new\_node->next = NULL;

        if(AdjacencyList == NULL)AdjacencyList = new\_node;

        else{

            nodearr\_ptr ptr = AdjacencyList;

            while(ptr->next != NULL)ptr = ptr->next;

            ptr->next = new\_node;

        }

    }

    int AdjacencyMatrix[nov][nov];

    for(int i=0;i<nov;i++)for(int j=0;j<nov;j++)AdjacencyMatrix[i][j] = 0;

    // Adding Nodes

    nodearr\_ptr ptr\_trav;ptr\_trav = AdjacencyList;

    for(int i=0; i<nov;i++){

        int deg;

        node\_ptr ptr1\_trav;

        printf("    Enter %d Node degree: ",i);scanf("%d",&deg);

        for(int j=0;j<deg;j++){

            int value;

            printf("        Enter %d Node %d Connection: ",i,j);scanf("%d",&value);

            node\_ptr new\_node = (node\_ptr) malloc(sizeof(node\_ptr));

            new\_node->data = value;

            new\_node->next = NULL;

            AdjacencyMatrix[i][value] = 1;

            if(j == 0){ptr1\_trav = new\_node;ptr\_trav->right = new\_node;}

            else{ptr1\_trav->next = new\_node;ptr1\_trav=new\_node;}

        }

        ptr\_trav = ptr\_trav->next;

    }

    //Printing Adjacency MAtrix

    printf("Adjacency Matrix: \n");

    for(int i=0;i<nov;i++){for(int j=0;j<nov;j++)printf("%d ",AdjacencyMatrix[i][j]);printf("\n");}

    //Printing Adjacency List

    printf("\n AdjacencyList: \n");

    nodearr\_ptr ptr;

    ptr = AdjacencyList;

    while(ptr != NULL){

        printf("%d",ptr->data);

        node\_ptr ptr1 = ptr->right;

        while(ptr1->next != NULL){

            printf(" -> %d",ptr1->data);

            ptr1 = ptr1->next;

        }

        printf(" -> %d \n",ptr1->data);

        ptr = ptr->next;

    }

}

int main(){

    GenerateGraph();

    return 0;

}

Output:

A computer screen shot of a black screen

Description automatically generated