VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

On

DATA STRUCTURES (23CS3PCDST)

Submitted by

Rohan Vats (1BM23CS273)

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

September 2024-January 2025

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

This is to certify that the Lab work entitled **"DATA STRUCTURES"** carried out by **Rohan Vats (1BM23CS273)**, who is Bonafide student of **B. M. S. College of Engineering**. It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2024-25. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab - **(23CS3PCDST)** work prescribed for the said degree.

Prof. Lakshmi Neelima M	**Dr. Kavitha Sooda**
Assistant Professor	Professor and Head
Department of CSE	Department of CSE
BMSCE, Bengaluru	BMSCE, Bengaluru

Index Sheet

Course outcomes:

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyse data structure operations for a given problem
CO3	Design and develop solutions using the operations of linear and nonlinear data structure for a given specification.
CO4	Conduct practical experiments for demonstrating the operations of different data structures.

Lab program 1

Write a program to simulate the working of stack using an array with the following:

a) Push

b) Pop

c) Display

The program should print appropriate messages for stack overflow, stack underflow.

Program:

```
#include<stdio.h>
#include<stdlib.h>
#define STACK_SIZE 5
void push (int st[],int *top)
{
        int item;
        if(*top==STACK_SIZE-1)
                printf("Stack overflow\n");
        else
        {
                printf("\nEnter an item :");
                scanf("%d",&item);
                (*top)++;
                st[*top]=item;
        }
}
void pop(int st[],int *top)
{
        if(*top==-1)
                printf("Stack underflow\n");
        else
        {
                printf("\n%d item was deleted",st[(*top)--]);
        }
}

void display(int st[],int *top)
```

```
{
        int i;
        if(*top==-1)
                printf("Stack is empty\n");
        for(i=0;i<=*top;i++)
                printf("%d\t",st[i]);
}
void main()
{
        int st[10],top=-1, c,val_del;
        while(1)
        {
                printf("\n1. Push\n2. Pop\n3. Display\n");
                printf("\nEnter your choice :");
                scanf("%d",&c);
                switch(c)
                {
                        case 1: push(st,&top);
                                break;
                        case 2: pop(st,&top);
                                break;
                        case 3: display(st,&top);
                                break;
                        default: printf("\nInvalid choice!!!");
                                exit(0);
                }
        }
}
```

Output:

```
PS D:\DS> cd "d:\DS\" ; if ($?) { gcc stack.c -o stack } ; if ($?) { .\stack }

1. Push
2. Pop
3. Display

Enter your choice :2
Stack underflow

1. Push
2. Pop
3. Display

Enter your choice :1

Enter an item :10

1. Push
2. Pop
3. Display

Enter your choice :1

Enter an item :20

1. Push
2. Pop
3. Display

Enter your choice :2

20 item was deleted
1. Push
2. Pop
```

3. Display

Enter your choice :1

Enter an item :20

1. Push
2. Pop
3. Display

Enter your choice :1

Enter an item :30

1. Push
2. Pop
3. Display

Enter your choice :3
10 20 30
1. Push
2. Pop
3. Display

Enter your choice :0

Invalid choice!!!
PS D:\DS>

Lab program 2

1.WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)

Program:

```
#include<stdio.h>
#include<string.h>
int index1=0, pos=0, top=-1,length;
char symbol, temp, infix[20], postfix[20], stack[20];
void push(char symbol);
char pop();
int pred(char symbol);
void infixtopostfix();
int main(){
   printf("Enter the infix expression: ");
   scanf("%s",infix);
   infixtopostfix();
   printf("\nIntfix expression: %s\n", infix);
   printf("\nPostfix expression: %s\n", postfix);
   return 0;
}

void infixtopostfix(){
   length = strlen(infix);
   while(index1<length){
      symbol = infix[index1];
      switch(symbol){
         case '(': {push(symbol);
            break;
         }
```

```
            case ')':{ temp = pop();
               while(temp!='('){
                  postfix[pos++] = temp;
                  temp = pop();
               }
               break;
            }
            case '+':
            case '-':
            case '*':
            case '/':{while(pred(stack[top])>=pred(symbol)){
                  temp=pop();
                  postfix[pos++]=temp;
               }
               push(symbol);
               break;
            }
            default : postfix[pos++]=symbol;
         }
         index1++;
      }
      while(top>0){
         temp = pop();
         postfix[pos++]=temp;
      }
   }

   void push(char symbol){
      top++;
      stack[top] = symbol;
   }
```

```
char pop(){
    char symb = stack[top];
    top--;
    return symb;
}

int pred(char symbol){
    switch(symbol){
        case '+':
        case '-':return 1;
        case '*':
        case '/':return 2;
        case '(':return 3;
    }
    return 0;
}
```

Output:

```
Enter the infix expression: a+(b*c-(e/f)*g)*h

Intfix expression: a+(b*c-(e/f)*g)*h

Postfix expression: ab*c*+e*f/-
```

Lab program 3

Majority Element – Leet code Program

Given an array nums of size n, return *the majority element*.

The majority element is the element that appears more than ⌊n / 2⌋ times. You may assume that the majority element always exists in the array.

Program:

```
class Solution {
public:
    int majorityElement(vector<int>& nums) {
        int n = nums.size();
        int count=0,candidate;
        for(int i=0; i<n; i++){
            if(count==0){
                candidate=nums[i];
                count=1;
            }
            else if (nums[i]==candidate){
                count++;
            }
            else {
                count--;
            }
        }
        return candidate;
    }
};
```

Output:

Input: nums = [3,2,3]

Output: 3

Input: nums = [2,2,1,1,1,2,2]

Output: 2

Lab program 4

3a) WAP to simulate the working of a queue of integers using an array. Provide the following operations:
Insert, Delete, Display The program should print appropriate messages for queue empty and queue overflow conditions.

Program:

```
#include<stdio.h>
# define MAX 3
int q[MAX], front=-1, rear=-1, val;
void insert(int val);
int delete();
void newqueue();
void display();

void main(){
   int n=0;
   printf("1-> Insert\n2->Delete\n3->Display\n4->New Queue\n5->Exit\n");
   while(n!=5){
      printf("\nEnter your choise:");
      scanf("%d",&n);
      printf("\n");
      switch (n)
      {
         case 1:{
            printf("Enter the number to be inserted:");
            scanf("%d",&val);
            printf("\n");
            insert(val);
            break;
         }
         case 2:{
```

```
            val=delete();
            if (val!=-1){
                printf("The deleted element is:%d\n\n",val);
            }
            break;
        }
        case 3:{
            display();
            break;
        }
        case 4:{
            newqueue();
            printf("New queue is created.\n");
            break;
        }
        case 5:{
            printf("Exiting...\n");
            break;
        }
        default:{
            printf("Wrong option!!\n");
        }
    }

  }
}
void insert(int val){
  if (front==-1&&rear==-1){
    front=0;
    rear=0;
    q[rear]=val;
    return;
```

```
    }
    else if(rear==MAX-1){
        printf("The Queue is full.\n");
        return;
    }
    else{
        q[++rear]=val;
        return;
    }
}
int delete(){
    if(front==-1||front>rear){
        printf("The queue is empty.\n");
        return -1;
    }
    else{
        val=q[front++];
    }
}
void newqueue(){
    rear=-1;
    front=-1;
    return;
}
void display(){
    if(rear==-1||front>rear){
        printf("Queue is empty.\n");
    }
    else{
        printf("The queue is:");
        for(int i=front;i<=rear;i++){
            printf("%d ",q[i]);
```

```
        }
        printf("\n\n");
    }
}
```

Output:

```
1-> Insert
2->Delete
3->Display
4->New Queue
5->Exit

Enter your choise:3

Queue is empty.

Enter your choise:2

The queue is empty.

Enter your choise:1

Enter the number to be inserted:10


Enter your choise:1

Enter the number to be inserted:20


Enter your choise:1
```

Enter the number to be inserted:30

Enter your choise:3

The queue is:10 20 30

Enter your choise:1

Enter the number to be inserted:40

The Queue is full.

Enter your choise:3

The queue is:10 20 30

Enter your choise:2

The deleted element is:10

Enter your choise:2

The deleted element is:20

Enter your choise:2

The deleted element is:30

Enter your choise:3

Queue is empty.

Enter your choise:5

Exiting...

3b) WAP to simulate the working of a circular queue of integers using an array. Provide the following operations:
Insert, Delete & Display The program should print appropriate messages for queue empty and queue overflow conditions.

Program:

```
#include<stdio.h>
#include<stdlib.h>
# define SIZE 5
int front=-1,rear=-1,val;
int queue[SIZE];
void enqueue(int i);
int dequeue();
void display();
int isfull(){
   if(front==(rear+1)%SIZE){
      printf("Queue is full.\n\n");
      return 1;
   }
```

```
    return 0;
}
int isempty(){
    if(front==-1&&rear==-1){
        printf("Queue is empty.\n\n");
        return 1;
    }
    return 0;
}


void main(){
    int choice=0;
    printf("\n1->Insert\n2->Delete\n3->Display\n4->Exit\n");
    while(choice!=4){
        printf("Enter your choice:");
        scanf("%d",&choice);
        printf("\n");
        switch(choice){
            case 1:{
                printf("Enter the number to insert:");
                scanf("%d",&val);
                printf("\n\n");
                enqueue(val);
                break;
            }
            case 2:{
                val=dequeue();
                if(val==-1){
                    break;
                }
                else{
```

```
                printf("The deleated element is:%d\n\n",val);
                break;
            }
        }
        case 3:{
            display();
            break;
        }
        case 4:{printf("******Exit*******");
            break;
            }

        default:printf("Invalid choice!!\n\n");
    }
  }
}
void enqueue(int i){
  if(isfull()){
    return;
  }
  else if(front==-1&&rear==-1){
    front=0;
    rear=0;
    queue[rear]=i;
    return;
  }
  else{
    rear=(rear+1)%SIZE;
    queue[rear]=val;
    return;
  }
}
```

```
int dequeue(){
    if(isempty()){
        return -1;
    }
    else{
        val=queue[front];
        if(front==rear){
            rear=-1;
            front=-1;
        }
        else{
            front=(front+1)%SIZE;
        }
        return val;
    }
}
void display(){
    if(isempty()){
        return;
    }
    else{
        int i;
        printf("The queue is:");
        for(i=front;i!=rear;i=(i+1)%SIZE){
            printf("%d  ",queue[i]);
        }
        printf("%d\n\n",queue[i]);
        return;
    }
}
```

Output:

1->Insert

2->Delete

3->Display

4->Exit

Enter your choice:2

Queue is empty.

Enter your choice:3

Queue is empty.

Enter your choice:1

Enter the number to insert:10

Enter your choice:2

The deleated element is:10

Enter your choice:1

Enter the number to insert:

10

Enter your choice:1

Enter the number to insert:20

Enter your choice:1

Enter the number to insert:30

Enter your choice:3

The queue is:10 20 30

Enter your choice:2

The deleated element is:10

Enter your choice:3

The queue is:20 30

Enter your choice:4

******Exit*******

Lab program 5

WAP to Implement Single Link List with following operations: Sort the linked list, Reverse the linked list, Concatenation of two linked lists.

Program:

```
#include <stdio.h>
#include <malloc.h>
struct node{
   int data;
   struct node *next;
};
struct node *start=NULL;
struct node *creat_ll(struct node *);
struct node *insert_beg(struct node *);
struct node *insert_end(struct node *);
struct node *insert_at(struct node *);
struct node *delete_beg(struct node *);
struct node *delete_end(struct node *);
struct node *delete_at(struct node *);
void delete(struct node *);


int main(){
   int choice;
   printf("\n1. Create Linked List\n2. Insert at Beginning\n3. Insert at End\n4. Insert at Position\n5. Delete from Beginning\n6. Delete from End\n7. Delete from Position\n8. Display Linked List\n9. Exit\n");
   do{
      printf("Enter your choice: ");
      scanf("%d",&choice);
      switch(choice){
         case 1:{start=creat_ll(start);
         break;
         }
```

```
            case 2:{start=insert_beg(start);
            break;
            }
            case 3:{start=insert_end(start);
            break;
            }
            case 4:{start=insert_at(start);
            break;
            }
            case 5:{start=delete_beg(start);
            break;
            }
            case 6:{start=delete_end(start);
            break;
            }
            case 7:{start=delete_at(start);
            break;
            }
            case 8:{display(start);
            break;
            }
        }
    }while(choice!=9);
}

struct node *creat_ll(struct node *start){
    int num;
    printf("Enter -1 to end.");
    printf("Enter the value of nood:");
    scanf("%d",&num);
    while(num!=-1){
        struct node *new_node=(struct node *)malloc(sizeof(struct node));
```

```
        new_node->data=num;
        new_node->next=NULL;
        if(start==NULL){
            start=new_node;
        }
        else{
            struct node *ptr=start;
            while(ptr->next!=NULL){
                ptr=ptr->next;
            }
            ptr->next=new_node;
        }
        printf("Enter the value of nood:");
        scanf("%d",&num);
    }
    return start;
}
struct node *insert_beg(struct node *start){
    int num;
    printf("Enter the value to insert at beginning:");
    scanf("%d",&num);
    struct node *new_node=(struct node *)malloc(sizeof(struct node));
    new_node->data=num;
    new_node->next=start;
    start=new_node;
    return start;
}

struct node *insert_end(struct node *start){
    int num;
    printf("Enter the value to insert at end:");
    scanf("%d",&num);
```

```
    struct node *new_node=(struct node *)malloc(sizeof(struct node));
    new_node->data=num;
    new_node->next=NULL;
    if(start==NULL){
        start=new_node;
    }
    else{
        struct node *ptr=start;
        while(ptr->next!=NULL){
            ptr=ptr->next;
        }
        ptr->next=new_node;
    }
    return start;
}

struct node *insert_at(struct node *start){
    int num,pos;
    printf("Enter the value to insert:");
    scanf("%d",&num);
    printf("Enter the position to insert:");
    scanf("%d",&pos);
    if(pos==0){
        start=insert_beg(start);
    }
    else{
        struct node *new_node=(struct node *)malloc(sizeof(struct node));
        new_node->data=num;
        new_node->next=NULL;
        struct node *ptr=start;
        for(int i=0;i<pos-1;i++){
            if(ptr==NULL){
```

```
            printf("Position out of range.");
            return start;
        }
        ptr=ptr->next;
    }
    new_node->next=ptr->next;
    ptr->next=new_node;
  }
  return start;
}

struct node *delete_beg(struct node *start){
  if(start==NULL){
    printf("List is empty.");
    return start;
  }
  struct node *ptr=start;
  start=start->next;
  printf("Deleted node is: %d\n",ptr->data);
  free(ptr);
  return start;
}

struct node *delete_end(struct node *start){
  if(start==NULL){
    printf("List is empty.");
    return start;
  }
  struct node *ptr=start;
  if(ptr->next==NULL){
    start=NULL;
    printf("Deleted node is: %d\n",ptr->data);
```

```
        free(ptr);
        return start;
    }
    while(ptr->next->next!=NULL){
        ptr=ptr->next;
    }
    struct node *temp=ptr->next;
    ptr->next=NULL;
    printf("Deleted node is: %d\n",temp->data);
    free(temp);
    return start;
}

struct node *delete_at(struct node *start){
    int pos;
    printf("Enter the position to delete:");
    scanf("%d",&pos);
    if(pos==0){
        start=delete_beg(start);
    }
    else{
        struct node *ptr=start;
        for(int i=0;i<pos-1;i++){
            if(ptr==NULL || ptr->next==NULL){
                printf("Position out of range.");
                return start;
            }
            ptr=ptr->next;
        }
        struct node *temp=ptr->next;
        ptr->next=ptr->next->next;
        printf("Deleted node is: %d\n",temp->data);
```

```
        free(temp);
        return start;
    }
    return start;
}

void display(struct node *start){
    struct node *ptr=start;
    if(ptr==NULL){
        printf("List is empty.");
        return;
    }
    printf("List is: ");
    while(ptr!=NULL){
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
```

Output:

1. Create Linked List
2. Insert at Beginning
3. Insert at End
4. Insert at Position
5. Delete from Beginning
6. Delete from End
7. Delete from Position
8. Display Linked List
9. Exit

Enter your choice: 1

Enter -1 to end.Enter the value of nood:10

Enter the value of nood:20

Enter the value of nood:30

Enter the value of nood:40

Enter the value of nood:50

Enter the value of nood:-1

Enter your choice: 2

Enter the value to insert at beginning:5

Enter your choice: 8

List is: 5 10 20 30 40 50

Enter your choice: 3

Enter the value to insert at end:60

Enter your choice: 8

List is: 5 10 20 30 40 50 60

Enter your choice: 4

Enter the value to insert:30

Enter the position to insert:3

Enter your choice: 8

List is: 5 10 20 30 30 40 50 60

Enter your choice: 5

Deleted node is: 5

Enter your choice: 8

List is: 10 20 30 30 40 50 60

Enter your choice: 6

Deleted node is: 60

Enter your choice: 8

List is: 10 20 30 30 40 50

Enter your choice: 7

Enter the position to delete:2

Deleted node is: 30

Enter your choice: 8

List is: 10 20 30 40 50

Enter your choice: 9

WAP to Implement Single Link List to simulate Stack & Queue Operations.

Program:

```
#include<stdio.h>
#include<malloc.h>
struct node{
   int data;
   struct node *next;
};
struct node *top=NULL;
void push();
int pop(struct node *top);
void display(struct node *top);

void main(){
   int choice;
   printf("Enter:\n1)Push \n2)Pop \n3)Display \n4)Exit\n");
   printf("Enter your choice:");
   scanf("%d",&choice);
   do{
      switch(choice){
         case 1:{
            push();
            break;
         }
         case 2:{
            int n=pop(top);
            if(n!=-1){
               printf("The poped element =%d\n",n);
            }
            break;
```

```
            }
            case 3:{
                display(top);
                break;
            }
        }
    printf("Enter your choice:");
    scanf("%d",&choice);
    }while(choice!=4);
}

void push(){
    int num;
    printf("Enter the element to push:");
    scanf("%d",&num);
    struct node *newnode=(struct node *)malloc(sizeof(struct node));
    newnode->data=num;
    newnode->next=top;
    top=newnode;
}

int pop(struct node *top){
    if(top==NULL){
        printf("the stack is empty!!\n");
        return -1;
    }
    struct node *ptr=top;
    int a=top->data;
    top=top->next;
    free(ptr);
    return a;
}
```

```
void display(struct node *top){
    if(top==NULL){
        printf("the stack is empty!!\n");
        return;
    }
    struct node *ptr=top;
    printf("The stack is : ");
    while(ptr!=NULL){
        printf("%d, ",ptr->data);
        ptr=ptr->next;
    }
    return;
}
```

Output:

```
Enter:
1)Push
2)Pop
3)Display
4)Exit
Enter your choice:2
the stack is empty!!
Enter your choice:3
the stack is empty!!
Enter your choice:1
Enter the element to push:10
Enter your choice:1
Enter the element to push:20
Enter your choice:
1
Enter the element to push:30
```

Enter your choice:3

The stack is : 30, 20, 10, Enter your choice:2

The poped element =30

Enter your choice:3

The stack is : 20, 10,

Program:

```
#include<stdio.h>
#include<malloc.h>
struct node {
   int data;
   struct node *next;
};
struct node *front=NULL,*rear=NULL;
void insert(struct node *rear);
int delete(struct node *front);
void display(struct node *front);

void main(){
   int choice;
   printf("Enter:\n1)Insert \n2)Delete \n3)Display \n4)Exit\n");
   printf("Enter your choice :");
   scanf("%d",&choice);
   do{
      switch(choice){
         case 1:{
            insert(rear);
            break;
         }
         case 2:{
```

```
            int temp=delete(front);
            if(temp!=-1){
                printf("Deleted element is %d\n",temp);
                break;
            }
            break;
        }
        case 3:{
            display(front);
            break;
        }
    }
    printf("Enter your choice :");
    scanf("%d",&choice);
    }while(choice!=4);
}

void insert(struct node *rear){
    int num;
    struct node *newnode=(struct node*)malloc(sizeof(struct node));
    printf("Enter the data to be inserted :");
    scanf("%d",&num);
    newnode->data=num;
    newnode->next=NULL;
    rear->next=newnode;
    rear=newnode;
}

int delete(struct node *front){
    if(front==NULL){
        printf("Queue is empty\n");
        return -1;
```

```
    }
    struct node *ptr=front;
    int n=ptr->data;
    front=front->next;
    free(ptr);
    return n;
}

void display(struct node *front){
    if(front==NULL){
        printf("Queue is empty\n");
        return;
    }
    struct node *ptr=front;
    printf("The queue is :");
    while(ptr!=NULL){
        printf("%d, ",ptr->data);
        ptr=ptr->next;
    }
    return;
}
```

Output:

```
Enter:
1)Insert
2)Delete
3)Display
4)Exit
Enter your choice :2
Queue is empty
```

Enter your choice :3

Queue is empty

Enter your choice :1

Enter the data to be inserted :10

Enter your choice :1

Enter the data to be inserted :20

Enter your choice :1

Enter the data to be inserted :30

Enter your choice :3

The queue is : 10, 20, 30

Enter your choice :2

Deleted element is :10

Enter your choice :3

The queue is : 20, 30

Enter your choice :4

Lab program 6

WAP to Implement doubly link list with primitive operations Create a doubly linked list Insert a new node to the left of the node. Delete the node based on a specific value Display the contents of the list2.

Program:

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

struct node {
   int data;
   struct node *next;
   struct node *prev;
};

struct node *start = NULL;
struct node *create_ll(struct node *start);
struct node *insert_l(struct node *start);
struct node *delete_node(struct node *start);
void display(struct node *start);

void main() {
   int choice;
   printf("\n1. Create Linked List\n2. Insert Node\n3. Delete Node\n4. Display Linked List\n5. Exit\n");
   do{
      printf("Enter your choice: ");
      scanf("%d",&choice);
      printf("\n");
      switch(choice){
         case 1:
            start = create_ll(start);
            break;
```

```
            case 2:
                start = insert_l(start);
                break;
            case 3:
                start = delete_node(start);
                break;
            case 4:
                display(start);
                break;
            case 5:
                printf("Exiting...\n");
                break;
            default:
                printf("Invalid choice!\n");
        }
    }while(choice!=5);
}

struct node *create_ll(struct node *start) {
    struct node *newnode, *ptr;
    int num;
    printf("Enter -1 to stop entering nodes.\n");
    printf("Enter the data for the new node: ");
    scanf("%d",&num);
    while(num!= -1) {
        newnode = (struct node *)malloc(sizeof(struct node));
        newnode->data = num;
        newnode->next = NULL;
        if(start == NULL) {
            newnode->prev = NULL;
            start = newnode;
        }
```

```
        else {
            ptr = start;
            while(ptr->next!= NULL)
                ptr = ptr->next;
            ptr->next = newnode;
            newnode->prev = ptr;
        }
        printf("Enter the data for the new node: ");
        scanf("%d",&num);
    }
    return start;
}

struct node *insert_l(struct node *start) {
    struct node *newnode, *ptr;
    int n1,n2;
    printf("Enter the data before which new node is to be inserted: ");
    scanf("%d",&n1);
    printf("Enter the data for the new node: ");
    scanf("%d",&n2);
    newnode = (struct node *)malloc(sizeof(struct node));
    newnode->data = n2;
    if(start->data == n1) {
        newnode->next = start;
        start->prev = newnode;
        newnode->prev = NULL;
        start = newnode;
    }
    else {
        ptr = start;
        while(ptr->next!= NULL && ptr->next->data!= n1)
            ptr = ptr->next;
```

```
        if(ptr->next!= NULL) {
            newnode->next = ptr->next;
            ptr->next->prev = newnode;
            newnode->prev = ptr;
            ptr->next = newnode;
        }
        else
            printf("Node with given data not found in the list.\n");
    }
    return start;
}

struct node *delete_node(struct node *start) {
    struct node *ptr;
    int num;
    printf("Enter the data of the node to be deleted: ");
    scanf("%d",&num);
    ptr=start;
    while(ptr->data!= num){
        ptr=ptr->next;
    }
    if(ptr==start) {
        start=ptr->next;
        ptr->next->prev = NULL;
        free(ptr);
        return start;
    }
    if(ptr->next==NULL) {
        ptr->prev->next = NULL;
        free(ptr);
        return start;
    }
```

```
    if(ptr==NULL){
        printf("Node with given data not found in the list.\n");
        return start;
    }
    ptr->prev->next = ptr->next;
    ptr->next->prev = ptr->prev;
    free(ptr);
    return start;
}

void display(struct node *start) {
    struct node *ptr;
    if(start==NULL){
        printf("List is empty.\n");
        return;
    }
    printf("List is: ");
    ptr=start;
    while(ptr!=NULL){
        printf("%d ",ptr->data);
        ptr=ptr->next;
    }
    printf("\n");
}
```

Output:

1. Create Linked List
2. Insert Node
3. Delete Node
4. Display Linked List
5. Exit

Enter your choice: 1

Enter -1 to stop entering nodes.

Enter the data for the new node: 10

Enter the data for the new node: 20

Enter the data for the new node: 30

Enter the data for the new node: 40

Enter the data for the new node: 50

Enter the data for the new node: -1

Enter your choice: 4

List is: 10 20 30 40 50

Enter your choice: 2

Enter the data before which new node is to be inserted: 30

Enter the data for the new node: 25

Enter your choice: 4

List is: 10 20 25 30 40 50

Enter your choice: 3

Enter the data of the node to be deleted: 50

Enter your choice: 4

List is: 10 20 25 30 40

Enter your choice: 5

Exiting...

Lab program 7

Palindrome linked list – Leet code Program

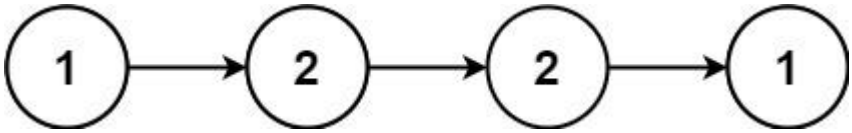
Given the head of a singly linked list, return true *if it is a palindrome* *or* false *otherwise.*

Program:

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
#include <stdbool.h>

struct ListNode* reverseList(struct ListNode* head) {
    struct ListNode* prev = NULL;
    struct ListNode* curr = head;
    struct ListNode* next = NULL;

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

    return prev;
}

bool isPalindrome(struct ListNode* head) {
    if (!head || !head->next) {
```

```
        return true;
    }
    struct ListNode* slow = head;
    struct ListNode* fast = head;
    while (fast && fast->next) {
        slow = slow->next;
        fast = fast->next->next;
    }

    struct ListNode* secondHalf = reverseList(slow);
    struct ListNode* firstHalf = head;
    struct ListNode* temp = secondHalf;
    bool isPalin = true;
    while (secondHalf) {
        if (firstHalf->val != secondHalf->val) {
            isPalin = false;
            break;
        }
        firstHalf = firstHalf->next;
        secondHalf = secondHalf->next;
    }
    reverseList(temp);

    return isPalin;
}
```

Output:



Input: head = [1,2,2,1]

Output: true

Lab program 8

Write a program

- **To construct a binary Search tree.**
- **To traverse the tree using all the methods i.e., in-order, preorder and post order**
- **To display the elements in the tree.**

Program:

```
#include<stdio.h>
#include<malloc.h>
typedef struct BST {
   int data;
   struct BST *left;
   struct BST *right;
}node;
node *createBST();
void insert(node *root,node *temp);
node *create();
void inorder(node *root);
void preorder(node *root);
void postorder(node *root);

void main(){
   node *root;
   root=createBST();
   printf("\nInorder Traversal of the BST is:\n");
   inorder(root);
   printf("\nPreorder Traversal of the BST is:\n");
   preorder(root);
   printf("\nPostorder Traversal of the BST is:\n");
   postorder(root);
}
```

```
node *createBST(){
    int i;
    node *root=NULL,*temp;
    do {
        temp=create();
        if(root==NULL){
            root=temp;
        }
        else{
            insert(root,temp);
        }
        printf("\n Enter '1' to stop inserting data or '0' to continue: ");
        scanf("%d",&i);
    }while(i==0);
    return root;
}

node *create(){
    node *temp;
    printf("\n enter data: ");
    temp=(node *)malloc(sizeof(node));
    scanf("%d",&temp->data);
    temp->left=temp->right=NULL;
    return temp;
}

void insert(node *root,node *temp){
    if(temp->data<root->data){
        if(root->left!=NULL){
            insert(root->left,temp);
        }
```

```
        else{
            root->left=temp;
        }
    }
    if(temp->data>root->data){
        if (root->right!=NULL){
            insert(root->right,temp);
        }
        else{
            root->right=temp;
        }
    }
}

void inorder(node *root){
    if(root!=NULL){
        inorder(root->left);
        printf("%d ",root->data);
        inorder(root->left);
    }
}

void preorder(node *root){
    if(root!=NULL){
        printf("%d ",root->data);
        preorder(root->left);
        preorder(root->right);
    }
}

void postorder(node *root){
    if(root!=NULL){
```

```
        postorder(root->left);
        postorder(root->right);
        rintf("%d ",root->data);
    }
}
```

Output:

```
enter data: 50

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 70

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 60

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 20

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 20

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 90

 Enter '1' to stop inserting data or '0' to continue: 0

 enter data: 10
```

Enter '1' to stop inserting data or '0' to continue: 0

enter data: 40

Enter '1' to stop inserting data or '0' to continue: 0

enter data: 100

Enter '1' to stop inserting data or '0' to continue: 1

Inorder Traversal of the BST is:

10 20 40 50 60 70 90 100

Preorder Traversal of the BST is:

50 20 10 40 70 60 90 100

Postorder Traversal of the BST is:

10 0 20 60 100 90 70 50

Lab program 9

Write a program to traverse a graph using BFS

Program:

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

#define MAX_VERTICES 100
typedef struct Queue {
    int items[MAX_VERTICES];
    int front;
    int rear;
} Queue;

void initQueue(Queue *q) {
    q->front = -1;
    q->rear = -1;
}
bool isQueueEmpty(Queue *q) {
    return q->front == -1;
}
void enqueue(Queue *q, int value) {
    if (q->rear == MAX_VERTICES - 1) {
        printf("Queue is full\n");
        return;
    }
    if (q->front == -1) {
        q->front = 0;
    }
    q->items[++q->rear] = value;
}
```

```
int dequeue(Queue *q) {
    if (isQueueEmpty(q)) {
        printf("Queue is empty\n");
        return -1;
    }
    int item = q->items[q->front];
    if (q->front == q->rear) {
        q->front = q->rear = -1;  // Queue is empty now
    } else {
        q->front++;
    }
    return item;
}

typedef struct Graph {
    int adjMatrix[MAX_VERTICES][MAX_VERTICES];
    int numVertices;
} Graph;
void initGraph(Graph *g, int vertices) {
    g->numVertices = vertices;
    for (int i = 0; i < vertices; i++) {
        for (int j = 0; j < vertices; j++) {
            g->adjMatrix[i][j] = 0;
        }
    }
}

void addEdge(Graph *g, int src, int dest) {
    g->adjMatrix[src][dest] = 1;
    g->adjMatrix[dest][src] = 1;
}
```

```
void bfs(Graph *g, int startVertex) {
    bool visited[MAX_VERTICES] = {false};
    Queue q;
    initQueue(&q);

    visited[startVertex] = true;
    enqueue(&q, startVertex);

    printf("BFS traversal starting from vertex %d: ", startVertex);
    while (!isQueueEmpty(&q)) {
        int vertex = dequeue(&q);
        printf("%d ", vertex);

        for (int i = 0; i < g->numVertices; i++) {
            if (g->adjMatrix[vertex][i] == 1 && !visited[i]) {
                visited[i] = true;
                enqueue(&q, i);
            }
        }
    }
    printf("\n");
}

int main() {
    Graph g;
    int vertices, edges, src, dest, startVertex;

    printf("Enter the number of vertices: ");
    scanf("%d", &vertices);

    initGraph(&g, vertices);
```

```
    printf("Enter the number of edges: ");
    scanf("%d", &edges);

    for (int i = 0; i < edges; i++) {
        printf("Enter edge %d (source destination): ", i + 1);
        scanf("%d %d", &src, &dest);
        addEdge(&g, src, dest);
    }

    printf("Enter the starting vertex for BFS: ");
    scanf("%d", &startVertex);

    bfs(&g, startVertex);

    return 0;
}
```

Output:

```
Enter the number of vertices: 7
Enter the number of edges: 8
Enter edge 1 (source destination): 0
1
Enter edge 2 (source destination): 0
2
Enter edge 3 (source destination): 0
3
Enter edge 4 (source destination): 0
4
Enter edge 5 (source destination): 1
5
Enter edge 6 (source destination): 3
```

5

Enter edge 7 (source destination): 2

6

Enter edge 8 (source destination): 4

6

Enter the starting vertex for BFS: 0

BFS traversal starting from vertex 0: 0 1 2 3 4 5 6

Lab program 10

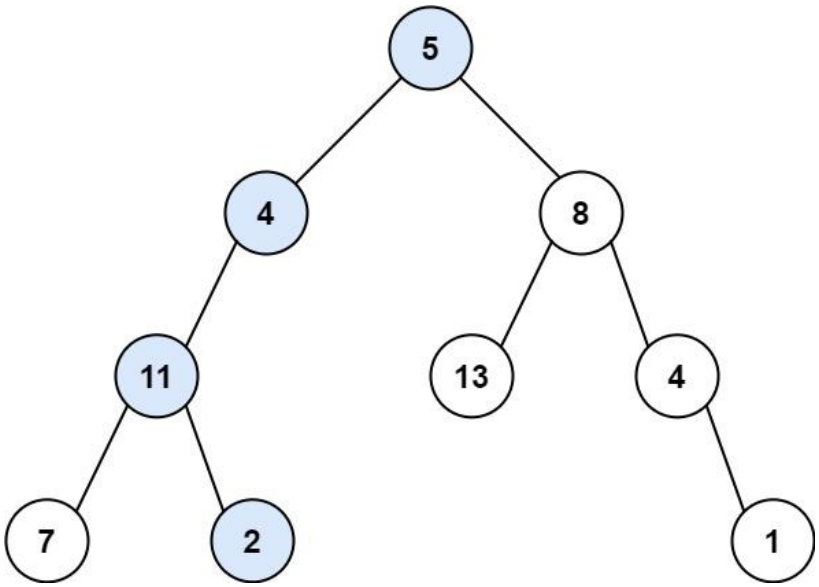
Path sum – Leet code Program

Given the root of a binary tree and an integer targetSum, return true if the tree has a root-to-leaf path such that adding up all the values along the path equals targetSum.

A leaf is a node with no children.

```
/**
 * Definition for a binary tree node.
 * struct TreeNode {
 *     int val;
 *     struct TreeNode *left;
 *     struct TreeNode *right;
 * };
 */
bool hasPathSum(struct TreeNode* root, int targetSum) {
    if (root == NULL) {
        return false;
    }

    if (root->left == NULL && root->right == NULL) {
        return root->val == targetSum;
    }

    return hasPathSum(root->left, targetSum - root->val) ||
           hasPathSum(root->right, targetSum - root->val);
}
```

Output:



Input: root = [1,2,3], targetSum = 5

Output: false

Explanation: There are two root-to-leaf paths in the tree:

(1 --> 2): The sum is 3.

(1 --> 3): The sum is 4.

There is no root-to-leaf path with sum = 5.