VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
On

DATA STRUCTURES (23CS3PCDST)

Submitted by

PREETHAM H D
(1BM23CS249)

in partial fulfillment 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 PREETHAM HD **(1BM23CS249)**, who is bonafide student of **B.M.S. College of Engineering**. It is in partial fulfillment 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
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Kavitha Sooda
Professor and Head
Department of CSE
BMSCE, Bengaluru

Index Sheet

Course outcomes:

CO1	Apply the concept of linear and nonlinear data structures.
CO2	Analyze 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.

```
#include<stdio.h>
#include<stdlib.h>
int stack[20];
int top=-1;
int N;
void push(int n);
int pop();
void display();
void main(){
  int num,choice;
  printf("enter the number of elements:");
  scanf("%d",&N);
  printf("enter the choice");
  printf("\n1:push:");
  printf("\n2.pop");
  printf("\n3.display");
  printf("\n4.exit");
  scanf("%d",&choice);
  while(choice!=4){
    switch(choice){
     case 1:{
       printf("\nenter the element:");
       scanf("%d",&num);
       push(num);
       display();
     break;
     }
    case 2:{
      int num1=pop();
    printf("\nthe deleted element is : %d",num1);
    display();
    break;
}
    case 3:{
    display();
    break;
    }
    case 4:{
    break;
    }
    default: break;
}
```

```
  printf("\nenter the choice:");
   scanf("%d",&choice);
}

}
void push(int n){
if(top==N-1){
    printf("\nStack Overflow");
}
else{
    stack[++top]=n;
}
}
int pop(){
    int ele;
if(top==-1){
    printf("\nStack Underflow");
}
else{
    ele=stack[top--];

}
return ele;
}
void display(){
    printf("\nthe stack elements are:");
for(int i=top;i>=0;i--){
    printf("\n %d",stack[i]);
}
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments> cd "c:\Users\preet\python_
UsingArr }
enter the number of elements:4
enter the choice
1:push:
2.pop
3.display
4.exit1

enter the element:23

the stack elements are:
 23
enter the choice:1

enter the element:45

the stack elements are:
 45
 23
enter the choice:1

enter the element:67

the stack elements are:
 67
 45
 23
enter the choice:1

enter the element:89
```

```
enter the element:90

Stack Overflow
the stack elements are:
 89
 67
 45
 23
enter the choice:2

the deleted element is : 89
the stack elements are:
the deleted element is : 89
the deleted element is : 89
the deleted element is : 89
the stack elements are:
 67
 45
 23
enter the choice:2

the deleted element is : 67
the stack elements are:
 45
 23
enter the choice:0

enter the choice:3

the stack elements are:
 45
 23
enter the choice:4
```

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)**

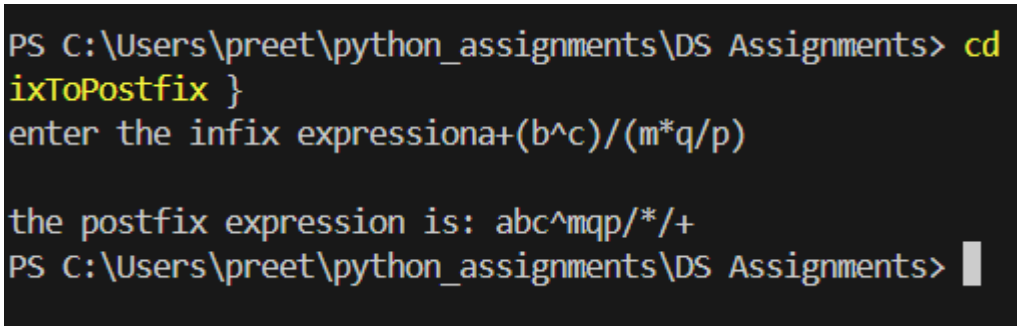
```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
int pos=0,top=-1,idx=0,len;
char symbol,stack[20],infix[20],postfix[20];
void push(char symb);
char pop();
int prece(char symbol);
void infixToPost();
void main(){
   printf("enter the infix expression");
   scanf("%s",infix);
   infixToPost();
   printf("\nthe postfix expression is: %s",postfix);
}
void infixToPost(){
   len=strlen(infix);
   push('#');
   while(idx<len){
      symbol=infix[idx];
      switch(symbol){

      case '^':
      case '*':
      case '/':
```

```
        case '+':
        case '-': { if(prece(stack[top])>prece(symbol)){
            char symb=pop();
            postfix[pos++]=symb;
            push(symbol);
        }
        else{
            push(symbol);
        } break;
        }
        case '(':{push(symbol);
                break;
        }
        case ')': {
                char symb=pop();
                while (symb!='('){
                    postfix[pos++]=symb;
                    symb=pop();
                }
                break;
        }
        default:{postfix[pos++]=symbol;
                 break;}
    }idx++;
    }
    while(stack[top]!='#'){
        postfix[pos++]=pop();
    }
}
```

```
void push(char symb){
   stack[++top]=symb;
}
char pop(){
   char s=stack[top--];
   return s;
}
int prece(char symb){
   int p;
   switch(symb){
      case '^': {
         p=3;
         break;
      }
      case '*': {
         p=2;
         break;
      }
      case '/': {
         p=2;
         break;
      }
      case '+': {
         p=1;
         break;
      }
      case '-': {
         p=1;
         break;
```

```
        }
        case '(':p=0;
        break;
        case '#': p=-1;
        break;
    }
    return p;
    }
```

Output:



Lab Program-3

a) **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**

```
#include<stdio.h>
#include<stdlib.h>
int size=3;
int lqueue[3];
int rear=-1,front=-1;
void enqueue(int data);
int dequeue();
void display();
int main(){
int choice;
while(1){
   printf("\n the choices are: ");
    printf("\n 1.insert an element  ");
     printf("\n 2.delete an element ");
      printf("\n 3.display the element ");
       printf("\n 4.exit ");
        printf("\n enter the choice: ");
       scanf("%d",&choice);
   switch(choice){
   case 1: {
      int data;
   printf("\n enter the element ");
   scanf("%d",&data);
   enqueue(data);
```

```
    break;
          }
    case 2: {
      int ele=dequeue();
      if(ele!=-1){
       printf("\n the deleted element is : %d",ele);
      }
      break;
          }
    case 3:{
          display();
          break;
          }
    case 4: exit(0);
    default: printf("\n enter valid choice");
    }
}
return 0;
}
void enqueue(int data){
if(rear==size-1){
    printf("\n queue overflow. ");
    return;
}
else if(rear==-1&&front==-1){
    front=0;
    lqueue[++rear]=data;
}
else{
```

```
    lqueue[++rear]=data;
}
}

int dequeue(){
if(front==-1){
    printf("\n the queue is empty");
    return -1;
}
else if (front==rear){
    int ele=lqueue[front];
    front=-1;
    rear=-1;
    return ele;
}
else{
    int ele=lqueue[front++];
    return ele;
}
}

void display(){
if(front==-1){
    printf("\n the queue is empty.");
}
else {
        printf("\n the queue elements are:.");
        int i=front;
    while(i<=rear){
```

```
        printf("\n %d",lqueue[i]);
        i++;
    }
}
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments> cd

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1

 enter the element 23

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1

 enter the element 67

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1

 enter the element 68
```

```
the queue elements are:.
67
68
the choices are:
1.insert an element
2.delete an element
3.display the element
4.exit
enter the choice: 2

the deleted element is : 67
the choices are:
1.insert an element
2.delete an element
3.display the element
4.exit
enter the choice: 2

the deleted element is : 68
the choices are:
1.insert an element
2.delete an element
3.display the element
4.exit
enter the choice: 2

the queue is empty
the choices are:
1.insert an element
2.delete an element
3.display the element
4.exit
enter the choice: 4
PS C:\Users\preet\python assignments\DS
```

b) 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

```
#include<stdio.h>
#include<stdlib.h>
int size=3;
int cqueue[3];
int rear=-1,front=-1;
void enqueue(int data);
int dequeue();
void display();
int main(){
int choice;
while(1){
    printf("\n the choices are: ");
     printf("\n 1.insert an element  ");
      printf("\n 2.delete an element ");
       printf("\n 3.display the element ");
        printf("\n 4.exit ");
         printf("\n enter the choice: ");
        scanf("%d",&choice);
   switch(choice){
   case 1: {
       int data;
   printf("enter the element ");
   scanf("%d",&data);
   enqueue(data);
   break;
          }
   case 2: {
```

```
        int ele=dequeue();
        if(ele!=-1){
         printf("the deleted element is : %d",ele);
        }
        break;
            }
    case 3:{
            display();
            break;
            }
    case 4: exit(0);
    default: printf("enter valid choice");
    }
}
return 0;
}
void enqueue(int data){
if((rear==size-1&&front==0)||((rear+1)%size==front)){
    printf("\n queue overflow. ");
    return;
}
else if(rear==-1&&front==-1){
    front=0;
    rear=0;
    cqueue[rear]=data;
}
else{
    rear=(rear+1)%size;
    cqueue[rear]=data;
```

```
}
}

int dequeue(){
if(front==-1&&rear==-1){
   printf("\n the queue is empty");
   return -1;
}
else if (front==rear){
   int ele=cqueue[front];
   front=-1;
   rear=-1;
   return ele;
}
else{
   int ele=cqueue[front];
   front=(front+1)%size;
   return ele;
}
}

void display(){
if(front==-1&&rear==-1){
   printf("\n the queue is empty.");
}
else {
   printf("\n the queue elements are:");
   int i=front;
   if(rear>front){
```

```
        while(i<=rear){
        printf("\n %d",cqueue[i]);
        i++;
    }
    } else{
    while(i>rear){
        printf("\n %d",cqueue[i]);
        i=(i+1)%size;
    }
    }
}
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments>
larQueue }

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1
enter the element 12

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1
enter the element 34

 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 1
enter the element 56
```

```
 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 2
the deleted element is : 34
 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 2
the deleted element is : 56
 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 2
the deleted element is : 78
 the choices are:
 1.insert an element
 2.delete an element
 3.display the element
 4.exit
 enter the choice: 2

 the queue is empty
```

Lab Program-4

a) **WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. Display the contents of the linked list.**

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node *Next;
};
struct Node * newNode(int data){
   struct Node* new;
   new= (struct Node *)malloc(sizeof(struct Node));
   new->data=data;
   new->Next=NULL;
   return new;
}
 struct Node * insert_beg(struct Node * start,int data){
      struct Node *newN;
      newN=newNode(data);
      newN->Next=start;
      start=newN;
      printf("\n inserted %d at beginning",data);
      return start;
}
struct Node * insert_end(struct Node * start,int data){
      struct Node *newN,*ptr;
      newN=newNode(data);
      ptr=start;
```

```
    while(ptr->Next!=NULL){
      ptr=ptr->Next;
    }
    ptr->Next=newN;
    printf("\n inserted %d at end",data);
    return start;
}
struct Node *insert_pos(struct Node *start,int data, int pos){
   struct Node *new,*ptr;
   new=newNode(data);
   int idx=1;
   ptr=start;
   if(pos==0){
      return insert_beg(start,data);
   }
   while(idx<pos-1&&ptr!=NULL){
   ptr=ptr->Next;
   idx++;
   }

   new->Next=ptr->Next;
   ptr->Next=new;
   printf("\n inserted %d at position %d",data,pos);
   return start;
}
void display(struct Node *start){
   struct Node *ptr;
   ptr=start;
   printf("\n");
```

```
    while(ptr!=NULL){
        printf("%d -> ",ptr->data);
        ptr=ptr->Next;
    }
    printf("NULL");
}
struct Node * delete_beg(struct Node * start){
    struct Node *ptr=start;
    start=start->Next;
    printf("\n %d is deleted from beginning",ptr->data);
    free(ptr);
    return start;
}
struct Node * delete_end(struct Node * start){
    struct Node *ptr,*n;
    n=start;
    ptr=start;
    while(ptr->Next->Next!=NULL){
        ptr=ptr->Next;
    }
    n=ptr->Next;
    ptr->Next=NULL;
    printf("\n %d is deleted from end",n->data);
    free(n);
    return start;
}
struct Node * delete_pos(struct Node * start,int pos){
    struct Node *ptr,*n;
    ptr=start;
```

```
    n=start;
    int idx=1;
    while(idx<pos-1&&ptr!=NULL){
    ptr=ptr->Next;
    idx++;
    }
    n=ptr->Next;
    ptr->Next=ptr->Next->Next;
    printf("\n %d is deleted from pos %d",n->data,pos);
    free(n);
    return start;
}
void main(){
    struct Node* start=NULL;
    start=insert_beg(start,10);
    start=insert_beg(start,20);
    start=insert_beg(start,50);
    start=insert_beg(start,70);
    start=insert_end(start,66);
    start=insert_pos(start,888,3);
    display(start);
    start=delete_end(start);
    start=delete_beg(start);
    display(start);
}
```

OUTPUT:

```
PS C:\Users\preet\python_assignments\DS Assignments> cd "

 inserted 10 at beginning
 inserted 20 at beginning
 inserted 50 at beginning
 inserted 70 at beginning
 inserted 66 at end
 inserted 888 at position 3
70 -> 50 -> 888 -> 20 -> 10 -> 66 -> NULL
 66 is deleted from end
 70 is deleted from beginning
50 -> 888 -> 20 -> 10 -> NULL
```

Lab Program-6

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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node *Next;
};
struct Node * newNode(int data){
   struct Node* new;
   new= (struct Node *)malloc(sizeof(struct Node));
   new->data=data;
   new->Next=NULL;
   return new;
}
 struct Node * insert_beg(struct Node * start,int data){
      struct Node *newN;
      newN=newNode(data);
      newN->Next=start;
      start=newN;
      printf("\n inserted %d at beginning",data);
      return start;
}
struct Node * insert_end(struct Node * start,int data){
      struct Node *newN,*ptr;
      newN=newNode(data);
      ptr=start;
      while(ptr->Next!=NULL){
```

```
        ptr=ptr->Next;
      }
      ptr->Next=newN;
      printf("\n inserted %d at end",data);
      return start;
}
struct Node *insert_pos(struct Node *start,int data, int pos){
    struct Node *new,*ptr;
    new=newNode(data);
    int idx=1;
    ptr=start;
    if(pos==0){
        return insert_beg(start,data);
    }
    while(idx<pos-1&&ptr!=NULL){
    ptr=ptr->Next;
    idx++;
    }

    new->Next=ptr->Next;
    ptr->Next=new;
    printf("\n inserted %d at position %d",data,pos);
    return start;
}
void display(struct Node *start){
    struct Node *ptr;
    ptr=start;
    printf("\n the linked list is: ");
    while(ptr!=NULL){
```

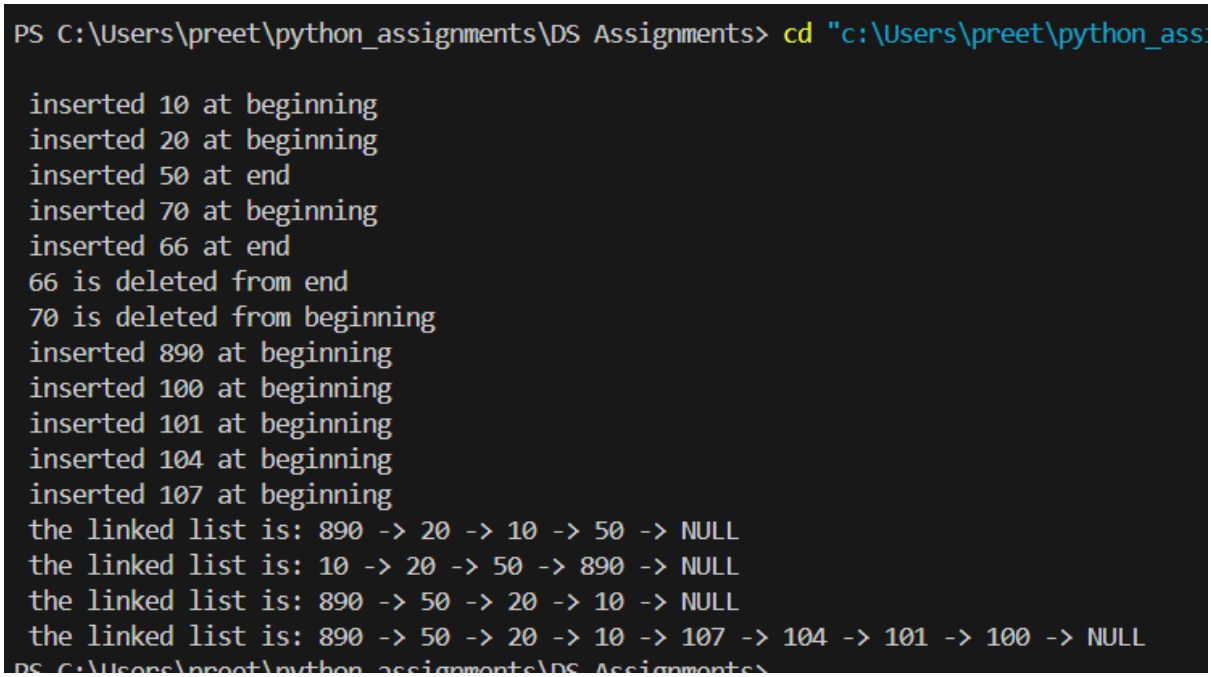
```
        printf("%d -> ",ptr->data);
        ptr=ptr->Next;
    }
    printf("NULL");
}
struct Node * delete_beg(struct Node * start){
    struct Node *ptr=start;
    start=start->Next;
    printf("\n %d is deleted from beginning",ptr->data);
    free(ptr);
    return start;
}
struct Node * delete_end(struct Node * start){
    struct Node *ptr,*n;
    n=start;
    ptr=start;
    while(ptr->Next->Next!=NULL){
        ptr=ptr->Next;
    }
    n=ptr->Next;
    ptr->Next=NULL;
    printf("\n %d is deleted from end",n->data);
    free(n);
    return start;
}
struct Node *sort(struct Node* start){
    struct Node *curr=start;
    while(curr!=NULL){
        struct Node* dup=curr->Next;
```

```
        while(dup!=NULL){
            if(curr->data>dup->data){
                int temp=dup->data;
                dup->data=curr->data;
                curr->data=temp;
            }
            dup=dup->Next;
            }
            curr=curr->Next;
        }



    return start;
}
struct Node *concat(struct Node*start, struct Node *start1){
    struct Node *ptr=start;
    while(ptr->Next!=NULL){
        ptr=ptr->Next;
    }
    ptr->Next=start1;


    return start;
}
struct Node *rev(struct Node*start){
    struct Node *prev=NULL;
    struct Node *curr=start;
    struct Node *next=curr->Next;
    while(next!=NULL){
        curr->Next=prev;
```

```
        prev=curr;
        curr=next;
        next=next->Next;
    }
    curr->Next=prev;
    return curr;
}
void main(){
    struct Node* start=NULL;
    struct Node *start1=NULL;
    start=insert_beg(start,10);
    start=insert_beg(start,20);
    start=insert_end(start,50);
    start=insert_beg(start,70);
    start=insert_end(start,66);
    start=delete_end(start);
    start=delete_beg(start);
    start=insert_beg(start,890);
    start1=insert_beg(start1,100);
    start1=insert_beg(start1,101);
    start1=insert_beg(start1,104);
    start1=insert_beg(start1,107);
    display(start);
    start=sort(start);
    display(start);
    struct Node *r=rev(start);
    display(r);
    struct Node *con=concat(r,start1);
     display(con);
```

}

OUTPUT:



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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node *Next;
};
struct Node * newNode(int data){
   struct Node* new;
   new= (struct Node *)malloc(sizeof(struct Node));
   new->data=data;
   new->Next=NULL;
```

```
    return new;
}
struct Node *push(struct Node *top,int data){
    struct Node *newN=newNode(data);
    if(top==NULL){
        top=newN;
    }
    else{
        newN->Next=top;
        top=newN;
    }
    printf("\n %d inserted",data);
    return top;
}
struct Node *pop(struct Node *top){
if(top==NULL){
    printf("underflow");
    return NULL;
}
struct Node *ptr=top;
top=top->Next;
printf("\n %d popped from stack",ptr->data);
free(ptr);
return top;
}
void display(struct Node *top){
    struct Node *ptr;
    ptr=top;
    printf("\n the stack is: ");
```

```
    while(ptr!=NULL){
        printf("%d -> ",ptr->data);
        ptr=ptr->Next;
    }
    printf("NULL");
}
void main(){
    struct Node *top=NULL;
    top=push(top,20);
    top=push(top,45);
    top=push(top,67);
    display(top);
    top=pop(top);
    display(top);
}
```

OUTPUT:

```
PS C:\Users\preet\python_assignments\DS Assignments> cd "c:\Us
ingLL }

 20 inserted
 45 inserted
 67 inserted
 the stack is: 67 -> 45 -> 20 -> NULL
 67 popped from stack
 the stack is: 45 -> 20 -> NULL
```

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   int data;
   struct Node *Next;
};
struct Node * newNode(int data){
   struct Node* new;
   new= (struct Node *)malloc(sizeof(struct Node));
   new->data=data;
   new->Next=NULL;
   return new;
}
struct Node *enqueue(struct Node *front,struct Node **rear,int data){
   struct Node *newN=newNode(data);
   if(front==NULL){
      front=newN;
      *rear=newN;
   }
   else{
      (*rear)->Next=newN;
      *rear=newN;
   }
   printf("\n %d inserted",data);
   return front;
}
struct Node *dequeue(struct Node *front){
if(front==NULL){
```

```
    printf("underflow");
    return NULL;
}
struct Node *ptr=front;
front=front->Next;
printf("\n %d removed from queue",ptr->data);
free(ptr);
return front;
}
void display(struct Node *front){
    struct Node *ptr;
    ptr=front;
    printf("\n The Queue is: ");
    while(ptr!=NULL){
        printf("%d -> ",ptr->data);
        ptr=ptr->Next;
    }
    printf("NULL");
}
void main(){
    struct Node *front=NULL;
    struct Node *rear=NULL;
    front=enqueue(front,&rear,20);
    front=enqueue(front,&rear,40);
    front=enqueue(front,&rear,50);
    front=enqueue(front,&rear,60);
    front=dequeue(front);
    display(front);
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments> cd
ingLL }

 20 inserted
 40 inserted
 50 inserted
 60 inserted20 removed from queue
 The Queue is: 40 -> 50 -> 60 -> NULL
```

Lab Program-7

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

```
#include<stdio.h>
#include<stdlib.h>
struct Node
{
   struct Node *prev;
   int data;
   struct Node *Next;
};
struct Node *newNode(int data){
   struct Node *new;
   new=(struct Node *)malloc(sizeof(struct Node));
   new->data=data;
   new->Next=NULL;
   new->prev=NULL;
   return new;
}
struct Node * insertleft(struct Node *start,int data,int val){
   struct Node *new=newNode(data);
   struct Node *ptr=start;
   if(start==NULL){
      start=new;
      return start;
   }
   while(ptr->data!=val){
      ptr=ptr->Next;
```

```
    }
    if(ptr->prev==NULL){

        new->Next=start;
        start->prev=new;
        start=new;
        return start;
    }
    ptr->prev->Next=new;
    new->prev=ptr->prev;
    new->Next=ptr;
    ptr->prev=new;
    return start;

}
struct Node *dele(struct Node *start,int val){
     struct Node *ptr=start;

     while(ptr->data!=val){
        ptr=ptr->Next;
     }
     if(ptr->prev==NULL&&ptr->Next==NULL){
        free(ptr);
        return NULL;
     }
     if(ptr->prev==NULL&&ptr->Next!=NULL){
        ptr->Next->prev=NULL;
        start=ptr->Next;
        free(ptr);
```

```
        return start;
    }
    if(ptr->prev!=NULL&&ptr->Next==NULL){
        ptr->prev->Next=NULL;
        free(ptr);
        return start;
    }
    (ptr->prev)->Next=ptr->Next;
    (ptr->Next)->prev=ptr->prev;
    free(ptr);

    return start;
}

void display(struct Node *start){
   struct Node *ptr=start;
   if(start==NULL){
       printf("the linked list is empty:");
   }
   else{
       printf("\n the linked lsit is:");
       while(ptr!=NULL){
           printf("%d ->",ptr->data);
           ptr=ptr->Next;
       }
       printf("NULL");
   }
}
void main(){
```

```
struct Node *start=NULL;
int choice;
printf("\n 1.insert left");
printf("\n 2.delete the specified element");
printf("\n 3.display");
printf("\n 4.exit");
printf("\n enter the choice:");
scanf("%d",&choice);
while(choice!=0){
   switch (choice)
   {
   case 1: {
      int data,val;
      printf("enter the data:");
      scanf("%d",&data);
       printf("enter the element :");
      scanf("%d",&val);
      start=insertleft(start,data,val);
      break;
   }
    case 2: {
      int data;
      printf("enter the element to be deleted :");
      scanf("%d",&data);
      start=dele(start,data);
      break;
   }
    case 3: {
       display(start);
```

```
        break;
    }
    case 4: exit(0);
    default:{printf("enter the valid choice");
        break;
    }}
     printf("\n enter the choice:");
scanf("%d",&choice);
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments> c

 1.insert left
 2.delete the specified element
 3.display
 4.exit
 enter the choice:1
enter the data:23
enter the element :1

 enter the choice:1
enter the data:45
enter the element :23

 enter the choice:1
enter the data:67
enter the element :23

 enter the choice:3

 the linked list is:45 ->67 ->23 ->NULL
 enter the choice:2
enter the element to be deleted :67

 enter the choice:3

 the linked list is:45 ->23 ->NULL
 enter the choice:4
```

Lab Program-8

Write a program a) To construct a binary Search tree. b) To traverse the tree using all the methods i.e., in order, preorder and post order

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

 struct Node{
    int data;
    struct Node *left;
    struct Node *right;
 };
  struct Node *newN(int data){
    struct Node *new=(struct Node *)malloc(sizeof(struct Node));
    new->data=data;
    new->left=NULL;
    new->right=NULL;
    return new;
  }
  struct Node * insert(struct Node *root,int data){
    if(root==NULL){
        return newN(data);
    }
    if(data<root->data){
        root->left=insert(root->left,data);
    }
    else{
        root->right=insert(root->right,data);
    }
```

```
  return root;
}

void inorder(struct Node* root){
  if(root==NULL){
      return;
  }
  inorder(root->left);
  printf("%d ",root->data);
  inorder(root->right);
}
void preorder(struct Node* root){
  if(root==NULL){
      return;
  }
  printf("%d ",root->data);
  preorder(root->left);
  preorder(root->right);
}
void postorder(struct Node* root){
  if(root==NULL){
      return;
  }
  postorder(root->left);
  postorder(root->right);
  printf("%d ",root->data);
}
void main(){
  struct Node *root=NULL;
```

```
 int choice;
printf("\n 1.insert");
printf("\n 2.display inorder");
printf("\n 3.display postorder");
printf("\n 4.display preorder");
printf("\n 5.exit");
printf("\n enter the choice:");
scanf("%d",&choice);
while(choice!=0){
   switch (choice)
   {
   case 1: {
      int data;
      printf("enter the data:");
      scanf("%d",&data);
      root=insert(root,data);
      break;
   }
    case 2: {
      printf("the inorder is:");
      inorder(root);
      break;
   }
    case 3: {
         printf("the postorder is:");
      postorder(root);
      break;
   }
    case 4: {
```

```
            printf("the preorder is:");
          preorder(root);
          break;
      }
      case 5: exit(0);

      default:{printf("enter the valid choice");
          break;
      }}
       printf("\n enter the choice:");
  scanf("%d",&choice);
  }
}
```

Output:

```
PS C:\Users\preet\python_assignments\DS Assignments>

 1.insert
 2.display inorder
 3.display postorder
 4.display preorder
 5.exit
 enter the choice:1
enter the data:34

 enter the choice:1
enter the data:23

 enter the choice:1
enter the data:37

 enter the choice:1
enter the data:45

 enter the choice:1
enter the data:78

 enter the choice:1
enter the data:90

 enter the choice:1
enter the data:58

 enter the choice:2
the inorder is:23 34 37 45 58 78 90
 enter the choice:3
the postorder is:23 58 90 78 45 37 34
 enter the choice:4
the preorder is:34 23 37 45 78 58 90
 enter the choice:5
```

Lab Program-9

a) Write a program to traverse a graph using BFS method.

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

void bfs(int adj[10][10],int n,int src){
   int q[10];
   int front=0,rear=-1;
   int visited[10]={0};
   int node;
   printf("the node visited from %d is:",src);
   q[++rear]=src;
   visited[src]=1;
   printf("%d->",src);
   while(front<=rear){
      int u=q[front++];
      for(int v=0;v<n;v++){
         if(adj[u][v]==1){
            if(visited[v]==0){
               printf(" %d ->",v);
               visited[v]=1;
               q[++rear]=v;
            }
         }
      }
   }
   printf("\n");
}
```

```
void main(){
    int n;
    int adj[10][10];
    int src;
    printf("enter the no of nodes:");
    scanf("%d",&n);
    printf("enter the adjacency matrix:");
    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            scanf("%d",&adj[i][j]);
        }
    }
    for(src=0;src<n;src++){
        bfs(adj,n,src);
    }
}
```

Output:

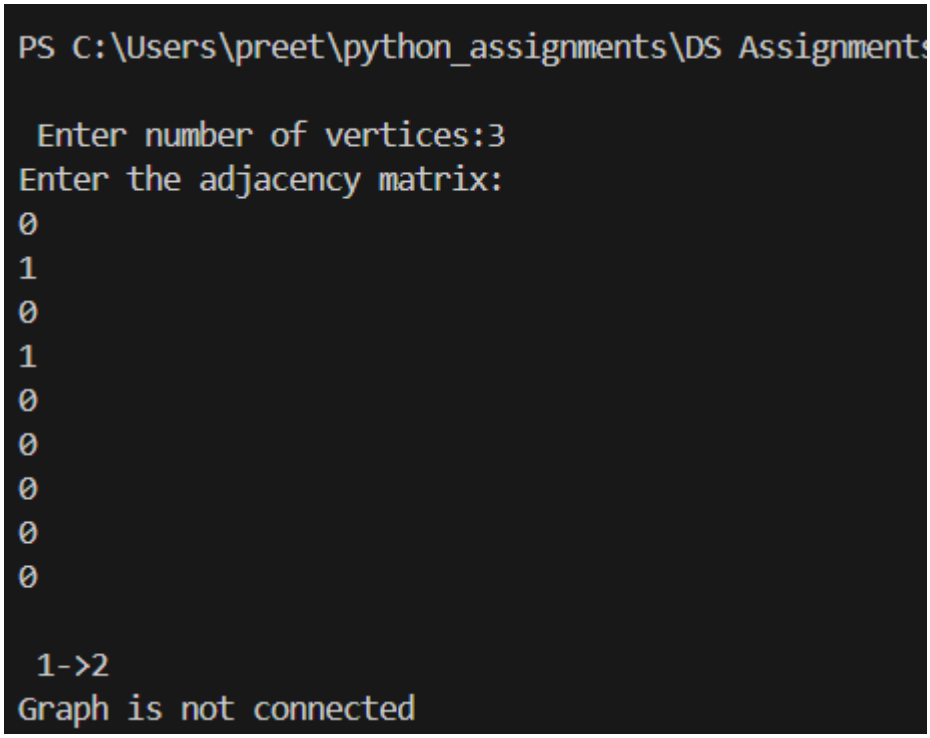
```
PS C:\Users\preet\python_assignments\DS Assignments> cd
enter the no of nodes:3
enter the adjacency matrix:0
1
1
1
0
1
1
1
0
the node visited from 0 is:0-> 1 -> 2 ->
the node visited from 1 is:1-> 0 -> 2 ->
the node visited from 2 is:2-> 0 -> 1 ->
```

b) Write a program to check whether given graph is connected or not using DFS method.

```
#include<stdio.h>
#include<stdlib.h>
int a[20][20], s[20], n;
void dfs(int v)
{
   int i;
   s[v]=1;
   for(i=1; i<=n; i++)
if(a[v][i] && !s[i])
{
   printf("\n %d->%d",v,i);
   dfs(i);
}
}
int main()
{
   int i, j, count=0;
   printf("\n Enter number of vertices:");
   scanf("%d", &n);
   for(i=1; i<=n; i++)
   {
s[i]=0;
for(j=1; j<=n; j++)
   a[i][j]=0;
   }
   printf("Enter the adjacency matrix:\n");
   for(i=1; i<=n; i++)
```

```
for(j=1; j<=n; j++)
    scanf("%d", &a[i][j]);
    dfs(1);
    printf("\n");
    for(i=1; i<=n; i++)
{
    if(s[i])
count++;
}
if(count==n)
    printf("Graph is connected");
else
    printf("Graph is not connected");
return 0;
}
```

Output:



Lab Program-10

Given a File of N employee records with a set K of Keys(4-digit) which uniquely determine the records in file F.Assume that file F is maintained in memory by a Hash Table (HT) of m memory locations with L as the set of memory addresses (2-digit) of locations in HT. Let the keys in K and addresses in L are integers. Design and develop a Program in C that uses Hash function H: K -> L as H(K)=K mod m (remainder method), and implement hashing technique to map a given key K to the address space L. Resolve the collision (if any) using linear probing.

```
#include <stdio.h>
#include<stdlib.h>
#define TABLE_SIZE 10
int h[TABLE_SIZE]={NULL};
void insert()
{
   int key,index,i,flag=0,hkey;
   printf("\nenter a value to insert into hash table\n");
   scanf("%d",&key);
   hkey=key%TABLE_SIZE;
   for(i=0;i<TABLE_SIZE;i++)
   {
    index=(hkey+i)%TABLE_SIZE;
      if(h[index] == NULL)
      {
      h[index]=key;
       break;
      }

   }
   printf("No of probes for %d is %d", key,i+1);
   if(i == TABLE_SIZE)
   printf("\nelement cannot be inserted\n");
```

```
}
void search()
{
int key,index,i,flag=0,hkey;
printf("\nenter search element\n");
scanf("%d",&key);
hkey=key%TABLE_SIZE;
    for(i=0;i<TABLE_SIZE; i++)
    {
     index=(hkey+i)%TABLE_SIZE;
        if(h[index]==key)
        {
            printf("value is found at index %d",index);
            break;
        }
    }
    if(i == TABLE_SIZE)
    printf("\n value is not found\n");
}
void display()
{
    int i;
printf("\nelements in the hash table are \n");
 for(i=0;i< TABLE_SIZE; i++)
 printf("\nat index %d \t value =  %d",i,h[i]);
}
int main()
{    int opt,i;
```

```
  while(1)
 {        printf("\nPress 1. Insert\t 2. Display \t3. Search \t4.Exit \n");
     scanf("%d",&opt);
   switch(opt)
   {
    case 1:insert();
           break;
    case 2:display();
           break;
   case 3:search();
           break;
   case 4:exit(0);
   }
 }
 return 0;
}
```

Output:

```
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
34
No of probes for 34 is 1
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
55
No of probes for 55 is 1
Press 1. Insert  2. Display     3. Search       4.Exit
1

enter a value to insert into hash table
79
No of probes for 79 is 1
Press 1. Insert  2. Display     3. Search       4.Exit
2

elements in the hash table are

at index 0       value =  0
at index 1       value =  0
at index 2       value =  0
at index 3       value =  0
at index 4       value =  34
at index 5       value =  55
at index 6       value =  0
at index 7       value =  0
at index 8       value =  0
at index 9       value =  79
Press 1. Insert  2. Display     3. Search       4.Exit
4
PS C:\Users\preet\python_assignments\DS Assignments>
```

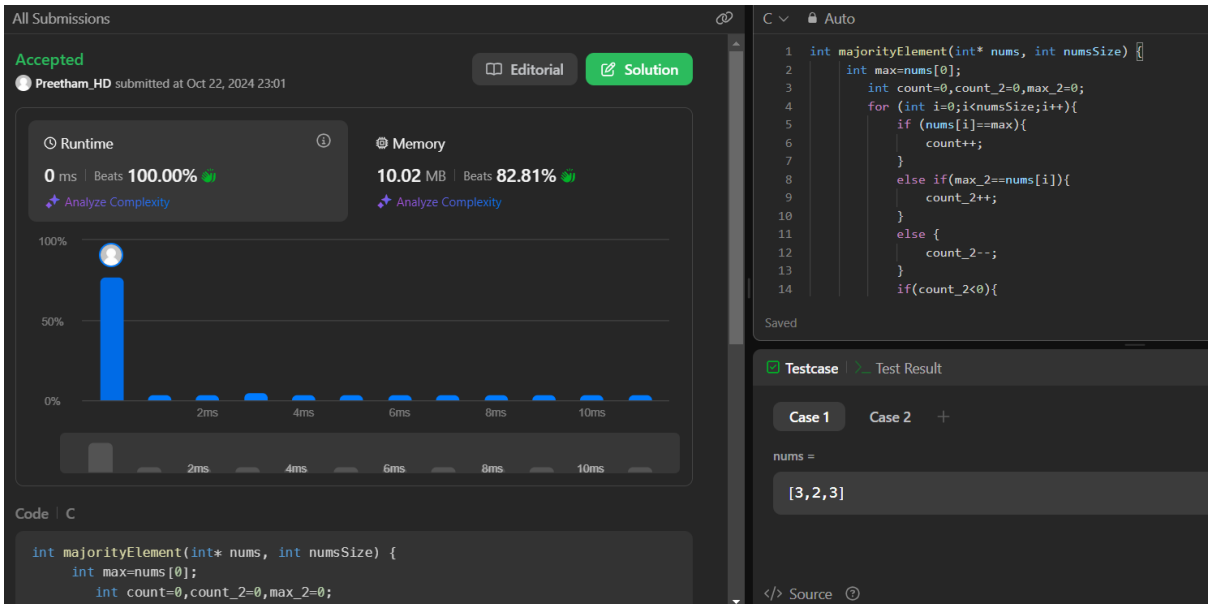
LEETCODE PROBLEMS

a)Majority element -169

```
int majorityElement(int* nums, int numsSize) {
    int max=nums[0];
      int count=0,count_2=0,max_2=0;
      for (int i=0;i<numsSize;i++){
          if (nums[i]==max){
              count++;
          }
          else if(max_2==nums[i]){
              count_2++;
          }
          else {
              count_2--;
          }
          if(count_2<0){
              max_2=nums[i];
              count_2=1;
          }
          if(count_2>count){
              int temp =max;
              max=max_2;
              max_2=temp;
              temp=count_2;
              count_2=count;
              count=temp;
          }
      }
      return max;
```

}

Output:

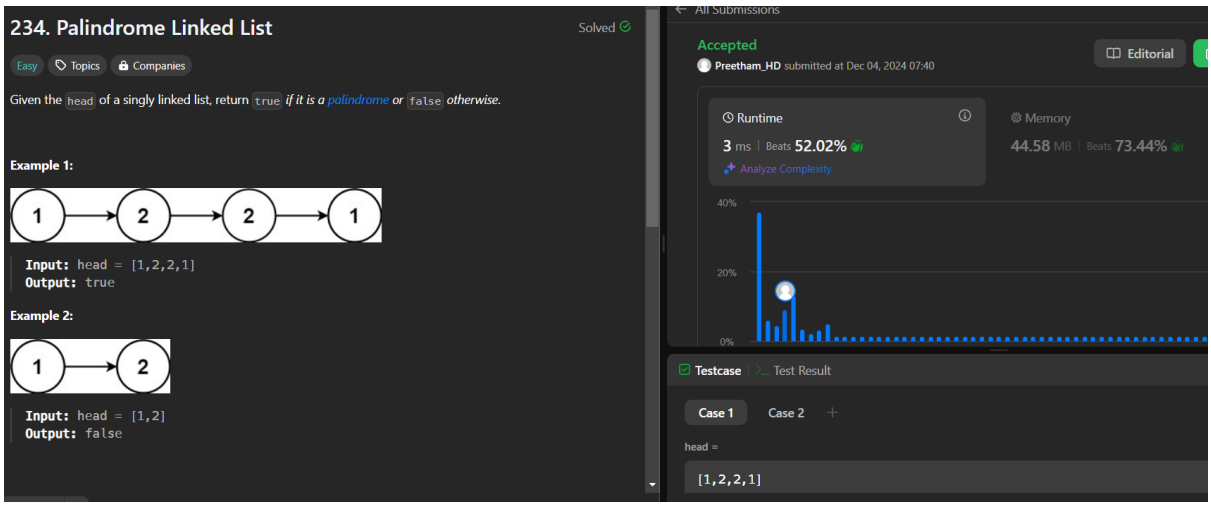


b**) LeetCode- is Palindrome**

```
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */


bool isPalindrome(struct ListNode* head) {
    if(head==NULL||head->next==NULL){
        return true;
    }

   struct ListNode *fast=head;
    struct ListNode *slow=head;
```

```
    while(fast!=NULL&&fast->next!=NULL){
        fast=fast->next->next;
        slow=slow->next;
    }
         struct ListNode *prev=NULL;
       struct ListNode *curr=slow;
        while(curr!=NULL){
            struct ListNode *Next=curr->next;
            curr->next=prev;
            prev=curr;
            curr=Next;
        }
     struct ListNode *frst=head;
     struct ListNode *sec=prev;
    while(sec!=NULL){
        if(frst->val!=sec->val){
            return false;
        }
        frst=frst->next;
        sec=sec->next;
    }
    return true;
}
```

Output:



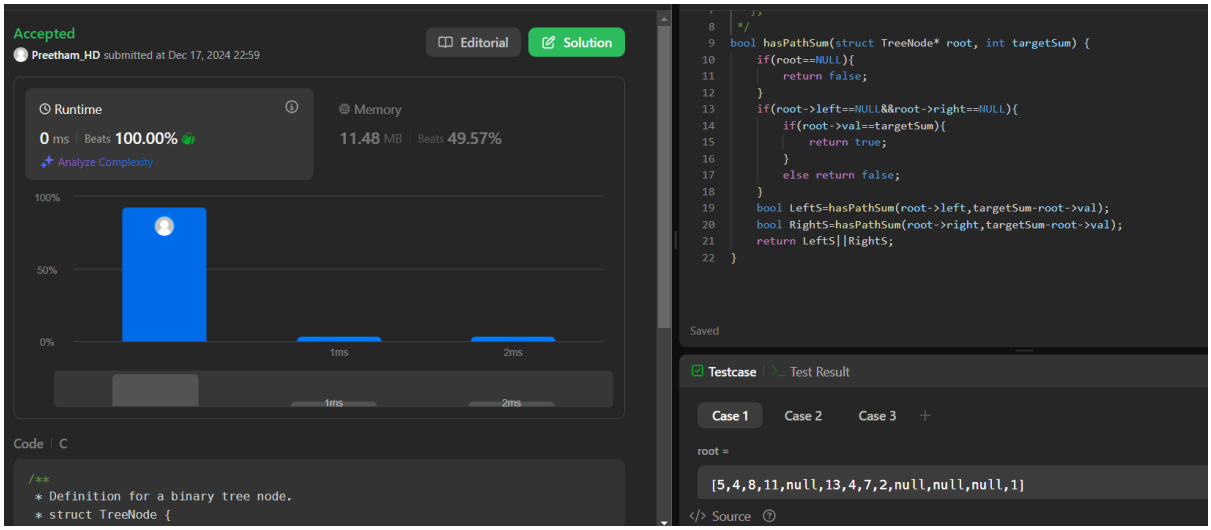
c) Pathsum

Leetcode-112

```
/**
 * 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){
        if(root->val==targetSum){
            return true;
        }
        else return false;
```

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

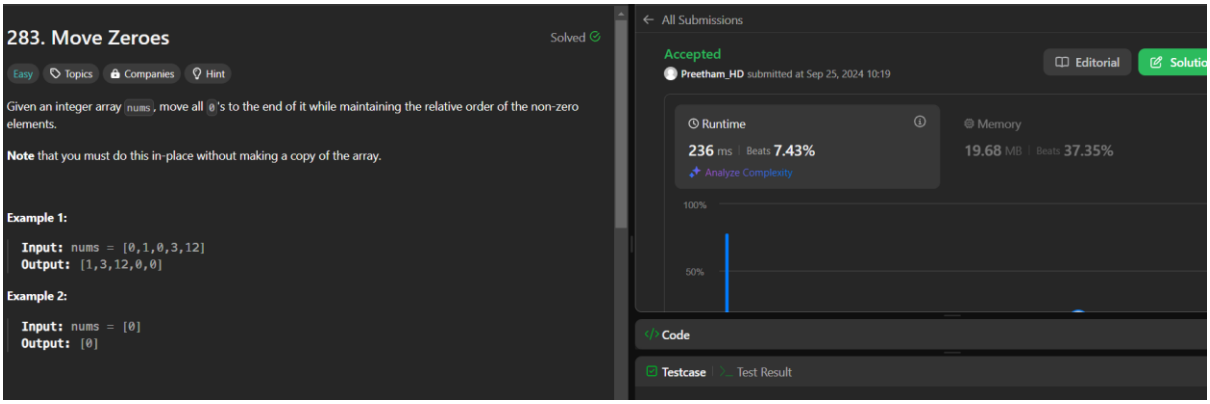
Output:



d) Move all zeroes
Leetcode-283

```
void moveZeroes(int* nums, int numsSize) {
    int l=0,r=numsSize-1;
    while(l<r){
     if(nums[l]==0){
        for(int i=l;i<r;i++){
            nums[i]=nums[i+1];
        }
        nums[r--]=0;

     }
     else l++;
    }

     }
```

Output:

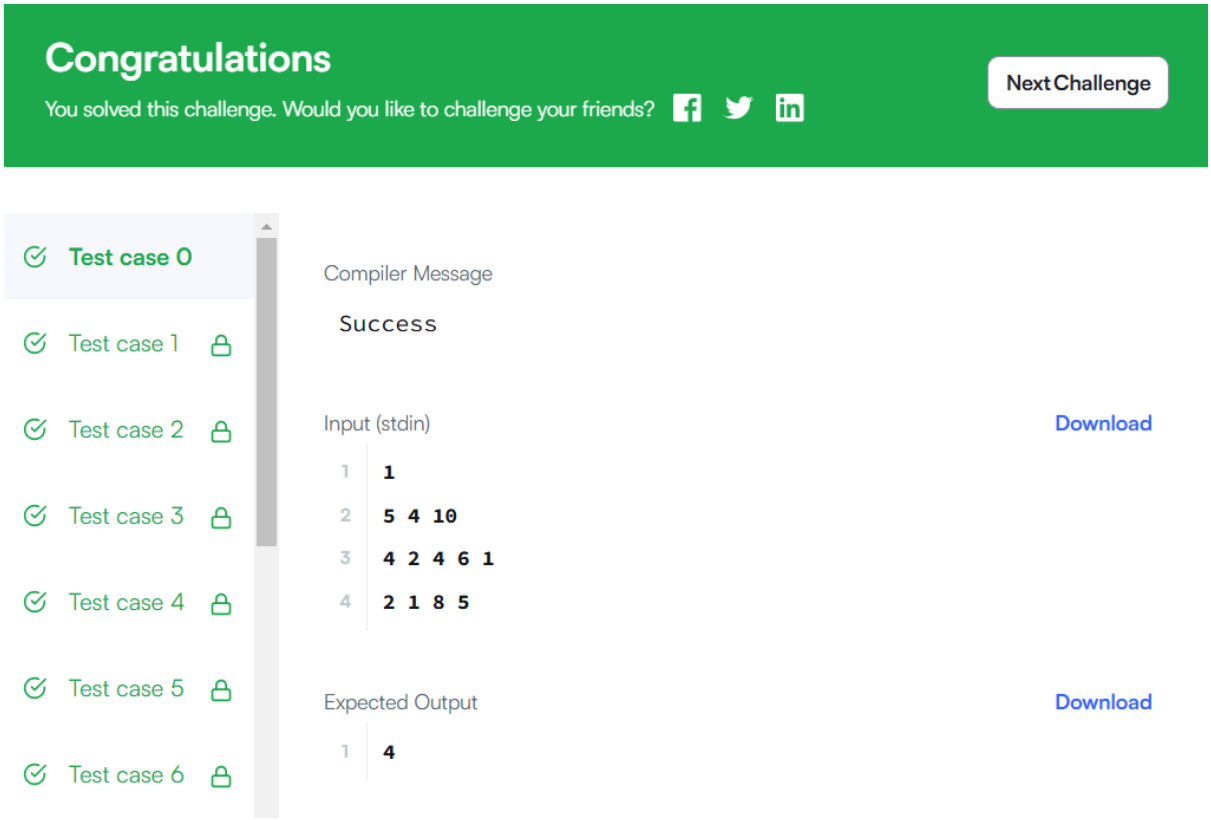


e) Hacker rank

Two stack Problem

```
int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
    int sum = 0, i = 0, j = 0, cnt = 0;
    while (i < a_count && sum + a[i] <= maxSum)
     { sum += a[i++];
      }
     cnt = i;
     while (i >= 0 && j < b_count) {
        sum += b[j++];
        while (sum > maxSum && i > 0)
         { sum -= a[--i]; }
          if (sum <= maxSum && (i + j) > cnt)
          { cnt = i + j;
           }
            }
    return cnt;

}
```

Output:


Congratulations
You solved this challenge. Would you like to challenge your friends?
Next Challenge
Test case 0
Test case 1
Test case 2
Test case 3
Test case 4
Test case 5
Test case 6
Compiler Message
Success
Input (stdin)
Download
1
5 4 10
4 2 4 6 1
2 1 8 5
Expected Output
Download
4