VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
On

DATA STRUCTURES (23CS3PCDST)

Submitted by
Rohan S Mirjankar
(1WN24CS232)

Under the guidance of
Prof. Manjula S
Assistant professor

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-January
2025-26

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 report entitled **“DATA STRUCTURES”** carried out by **Rohan S Mirjankar (1WN24CS232)**, 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 2025- 2026. 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.

Mrs. Manjula S
Assistant Professor
Department of CSE
BMSCE, Bengaluru

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

INDEX

Github: https://github.com/rohans-cs24/DS-lab

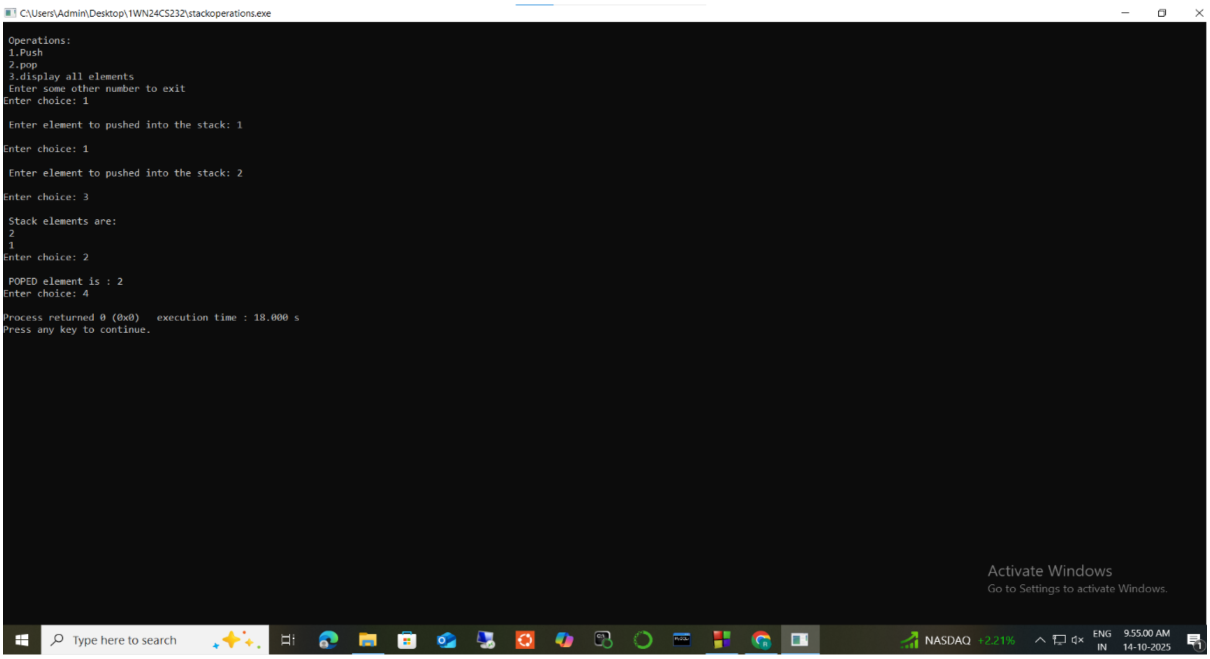
Course outcomes:

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.

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>
#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


C:\Users\Admin\Desktop\1WN24CS232\stackoperations.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
NASDAQ +2.21%
ENG
IN
9:55:00 AM
14-10-2025

```
 Operations:
 1.Push
 2.pop
 3.display all elements
 Enter some other number to exit
Enter choice: 1

 Enter element to pushed into the stack: 1

Enter choice: 1

 Enter element to pushed into the stack: 2

Enter choice: 3

 Stack elements are:
 2
 1
Enter choice: 2

 POPED element is : 2
Enter choice: 4

Process returned 0 (0x0)   execution time : 18.000 s
Press any key to continue.
```

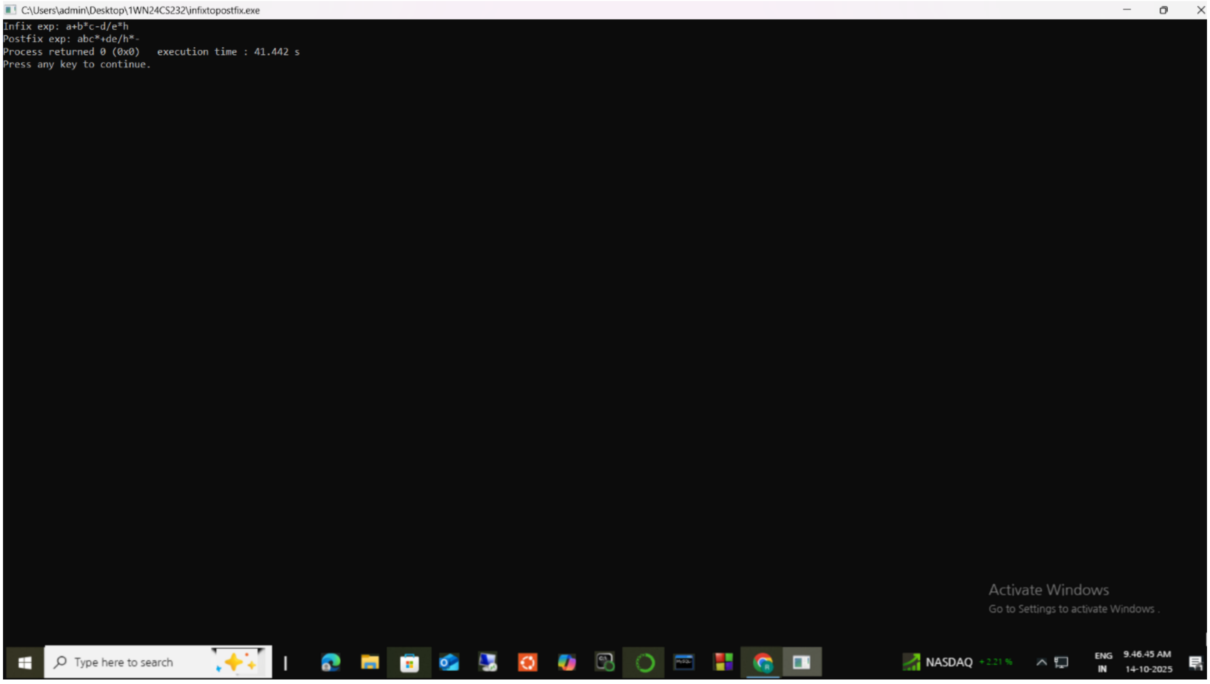
PROGRAM 2:

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 <ctype.h>
#define Max 10;
char stack[Max];
int top = -1;
void push(char ch)
{
   stack[++top] = ch;
}
char pop()
{
   return stack[top--];
}
int precedence(char ch)
{
   if (ch == '^')
      return 3;
   if (ch == '/' || ch == '*')
      return 2;
   if (ch == '-' || ch == '+')
      return 1;
   return 0;
}
void fun(char infix[])
{
   int j = 0;
   char postfix[Max];
   for (int i = 0; infix[i] != '\0'; i++)
   {
      char ch = infix[i];
      if (isalnum(ch))
      {
         postfix[j++] = ch;
      }
      else
      {
         if (ch == '(')
         {
            push(ch);
         }
         else
         {
            if (ch == ')')
            {
               while (top != -1 && stack[top] != '(')
                  postfix[j++] = pop();
               pop();
            }
            else
            {
               while (top != -1 && precedence(stack[top]) >= precedence(ch))
```

```
                        postfix[j++] = pop();
                    push(ch);
                }
            }
        }
    }
    while (top != -1)
    {
        postfix[j++] = pop();
    }
    postfix[j] = '\0';
    printf("Postfix exp: %s", postfix);
}
int main()
{
    char infix[Max];
    printf("Infix exp: ");
    scanf("%s", infix);
    fun(infix);
    return 0;
}
```

OUTPUT:


C:\Users\admin\Desktop\1WN24CS232\infixtopostfix.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
NASDAQ
ENG IN
9:46:45 AM
14-10-2025

```
Infix exp: a+b*c-d/e*h
Postfix exp: abc*+de/h*-
Process returned 0 (0x0)   execution time : 41.442 s
Press any key to continue.
```

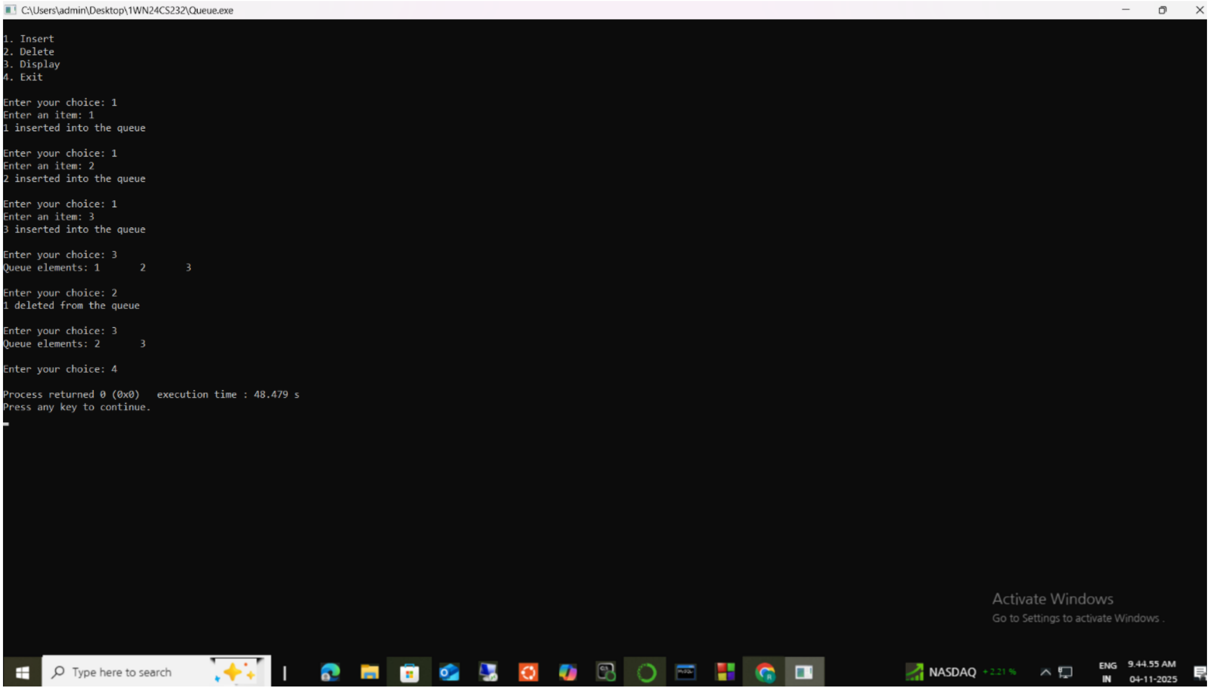
PROGRAM 3a:

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

```
#include <stdio.h>
#include <stdlib.h>
#define QUEUE_SIZE 5
int queue[QUEUE_SIZE], front = -1, rear = -1;
void insert(int queue[], int *front, int *rear)
{
   int item;
   if (*rear == QUEUE_SIZE - 1)
   {
      printf("Queue overflow\n");
   }
   else
   {
      printf("Enter an item: ");
      scanf("%d", &item);
      if (*front == -1)
      {
         *front = 0;
      }
      queue[++(*rear)] = item;
      printf("%d inserted into the queue\n", item);
   }
}
void delete(int queue[], int *front, int *rear)
{
   if (*front == -1 || *front > *rear)
   {
      printf("Queue is empty\n");
   }
   else
   {
      printf("%d deleted from the queue\n", queue[(*front)++]);
      if (*front > *rear)
      {
         *front = *rear = -1;
      }
   }
}
void display(int queue[], int *front, int *rear)
{
   if (*front == -1 || *front > *rear)
   {
      printf("Queue is empty\n");
   }
   else
   {
      printf("Queue elements: ");
      for (int i = *front; i <= *rear; i++)
      {
         printf("%d\t", queue[i]);
      }
```

```
        printf("\n");
    }
}
void main()
{
    int choice;
    printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    while (1)
    {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            insert(queue, &front, &rear);
            break;
        case 2:
            delete(queue, &front, &rear);
            break;
        case 3:
            display(queue, &front, &rear);
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice!!!\n");
        }
    }
}
```

OUTPUT:


C:\Users\admin\Desktop\1WN24CS232\Queue.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
NASDAQ
ENG
IN
9:44:55 AM
04-11-2025

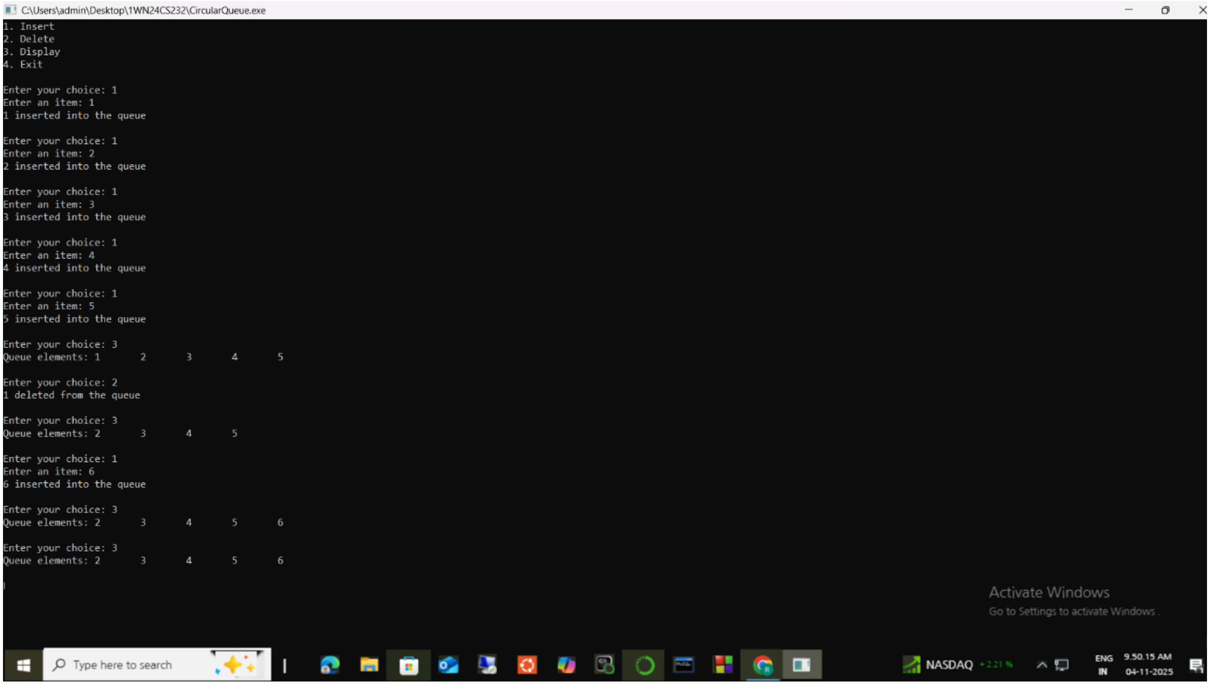
```
1. Insert
2. Delete
3. Display
4. Exit

Enter your choice: 1
Enter an item: 1
1 inserted into the queue

Enter your choice: 1
Enter an item: 2
2 inserted into the queue

Enter your choice: 1
Enter an item: 3
3 inserted into the queue

Enter your choice: 3
Queue elements: 1       2       3

Enter your choice: 2
1 deleted from the queue

Enter your choice: 3
Queue elements: 2       3

Enter your choice: 4

Process returned 0 (0x0)   execution time : 48.479 s
Press any key to continue.
```

PROGRAM 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

```
#include <stdio.h>
#include <stdlib.h>
#define QUEUE_SIZE 5
int queue[QUEUE_SIZE], front = -1, rear = -1;
void insert(int queue[], int *front, int *rear)
{
   int item;
   if ((*rear + 1) % QUEUE_SIZE == *front)
   {
      printf("Queue overflow\n");
   }
   else
   {
      printf("Enter an item: ");
      scanf("%d", &item);
      if (*front == -1)
      {
         *front = 0;
      }
      *rear = (*rear + 1) % QUEUE_SIZE;
      queue[*rear] = item;
      printf("%d inserted into the queue\n", item);
   }
}
void delete(int queue[], int *front, int *rear)
{
   if (*front == -1)
   {
      printf("Queue is empty\n");
   }
   else
   {
      printf("%d deleted from the queue\n", queue[*front]);
      if (*front == *rear)
      {
         *front = *rear = -1;
      }
      else
      {
         *front = (*front + 1) % QUEUE_SIZE;
      }
   }
}
void display(int queue[], int *front, int *rear)
{
   if (*front == -1)
   {
      printf("Queue is empty\n");
   }
   else
```

```
    {
        printf("Queue elements: ");
        int i = *front;
        while (1)
        {
            printf("%d\t", queue[i]);
            if (i == *rear)
            {
                break;
            }
            i = (i + 1) % QUEUE_SIZE;
        }
        printf("\n");
    }
}
void main()
{
    int choice;
    printf("\n1. Insert\n2. Delete\n3. Display\n4. Exit\n");
    while (1)
    {
        printf("\nEnter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            insert(queue, &front, &rear);
            break;
        case 2:
            delete(queue, &front, &rear);
            break;
        case 3:
            display(queue, &front, &rear);
            break;
        case 4:
            exit(0);
        default:
            printf("Invalid choice!!!\n");
        }
    }
}
```

OUTPUT:



C:\Users\admin\Desktop\1WN24CS232\CircularQueue.exe
Activate Windows
Go to Settings to activate Windows.
Type here to search
NASDAQ +2.21 %
ENG
IN
9.50.15 AM
04-11-2025

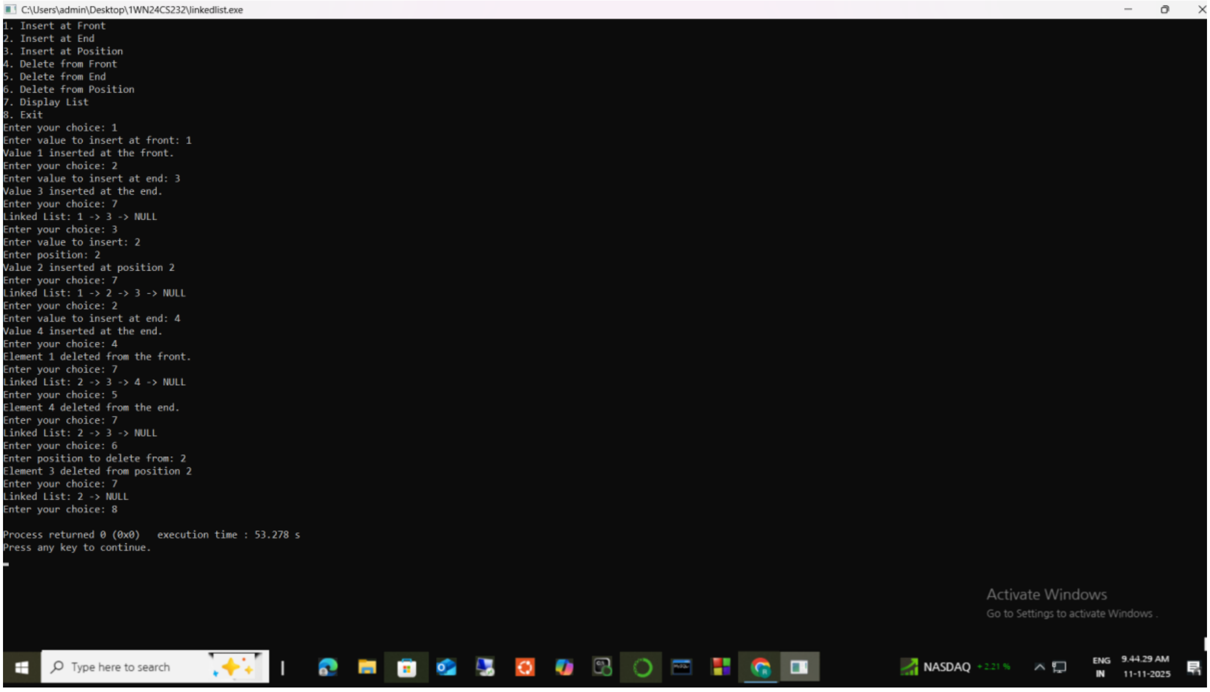
```
1. Insert
2. Delete
3. Display
4. Exit

Enter your choice: 1
Enter an item: 1
1 inserted into the queue

Enter your choice: 1
Enter an item: 2
2 inserted into the queue

Enter your choice: 1
Enter an item: 3
3 inserted into the queue

Enter your choice: 1
Enter an item: 4
4 inserted into the queue

Enter your choice: 1
Enter an item: 5
5 inserted into the queue

Enter your choice: 3
Queue elements: 1       2       3       4       5

Enter your choice: 2
1 deleted from the queue

Enter your choice: 3
Queue elements: 2       3       4       5

Enter your choice: 1
Enter an item: 6
6 inserted into the queue

Enter your choice: 3
Queue elements: 2       3       4       5       6

Enter your choice: 3
Queue elements: 2       3       4       5       6
```

PROGRAM 4:
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* head = NULL;
void insertFront(int value){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = head;
    head = newNode;
    printf("Value %d inserted at the front.\n", value);
}
void insertEnd(int value){
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;
    if(head == NULL){
        head = newNode;
        printf("Value %d inserted as the first node.\n", value);
        return;
    }
    struct node* temp = head;
    while(temp->next != NULL){
        temp = temp->next;
    }
    temp->next = newNode;
    printf("Value %d inserted at the end.\n", value);
}
void insertPosition(int value,int pos){
    if(pos<1){
        printf("Invalid position %d \n",pos);
        return;
    }
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    if(head == NULL || pos == 1){
        newNode->next = head;
        head = newNode;
        printf("Value %d inserted at position %d \n",value,pos);
        return;
    }
    struct node* temp = head;
    for(int i = 1;i < pos-1 && temp != NULL;i++){ // Corrected loop condition: temp->next is not
needed here
        temp = temp->next;
    }
    if(temp == NULL){
        printf("Invalid position..... %d\n",pos);
        free(newNode);
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
    printf("Value %d inserted at position %d \n",value,pos);
}
void deleteFront(){
    struct node* temp;
    if(head == NULL){
        printf("List is empty. Cannot delete.\n");
        return;
    }
    temp = head;
    head = temp->next;
    printf("Element %d deleted from the front.\n", temp->data);
    free(temp);
}

void deleteEnd(){
    if(head == NULL){
        printf("List is empty. Cannot delete.\n");
        return;
    }
    if(head->next == NULL){ // Only one node
        printf("Element %d deleted from the end.\n", head->data);
        free(head);
        head = NULL;
        return;
    }
    struct node* temp = head;
    while(temp->next->next != NULL){
        temp = temp->next;
    }
    printf("Element %d deleted from the end.\n", temp->next->data);
    free(temp->next);
    temp->next = NULL;
}
void deletePosition(int pos){
    if(pos<1){
        printf("Invalid position %d \n",pos);
        return;
    }
    if(head == NULL){
        printf("List is empty \n");
        return;
    }
    struct node* temp = head;
    if(pos == 1){
        head = temp->next;
        printf("Element %d deleted from position %d\n",temp->data,pos);
        free(temp);
        return;
    }
    for(int i= 1;i<pos-1 && temp!=NULL;i++){ // Corrected loop condition: temp->next is not needed
```

```
here
        temp = temp->next;
    }
    // Check if we reached the node *before* the deletion point
    if(temp == NULL || temp->next == NULL){
        printf("Invalid position.... %d\n",pos);
        return;
    }
    struct node* delNode = temp->next;
    temp->next = delNode->next;
    printf("Element %d deleted from position %d\n",delNode->data,pos);
    free(delNode);
}
void display(){
    if(head == NULL){
        printf("List is empty\n");
        return;
    }
    struct node* temp = head;
    printf("Linked List: ");
    while(temp != NULL){
        printf("%d -> ",temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}

int main(){
    int choice, value, pos;
        printf("1. Insert at Front\n");
        printf("2. Insert at End\n");
        printf("3. Insert at Position\n");
        printf("4. Delete from Front\n");
        printf("5. Delete from End\n");
        printf("6. Delete from Position\n");
        printf("7. Display List\n");
        printf("8. Exit\n");
    while(1){
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch(choice){
            case 1:
                printf("Enter value to insert at front: ");
                scanf("%d", &value);
                insertFront(value);
                break;
            case 2:
                printf("Enter value to insert at end: ");
                scanf("%d", &value);
                insertEnd(value);
                break;
            case 3:
                printf("Enter value to insert: ");
                scanf("%d", &value);
                printf("Enter position: ");
                scanf("%d", &pos);
                insertPosition(value, pos);
                break;
            case 4:
                deleteFront();
                break;
            case 5:
                deleteEnd();
                break;
            case 6:
                printf("Enter position to delete from: ");
                scanf("%d", &pos);
                deletePosition(pos);
                break;
            case 7:
                display();
                break;
            case 8:
                struct node* current = head;
                struct node* nextNode;
                while(current != NULL){
                    nextNode = current->next;
                    free(current);
                    current = nextNode;
                }
                head = NULL;
                return 0;
            default:
                printf("Invalid choice. Please enter a number between 1 and 8.\n");
        }
    }
    return 0;
}
```

OUTPUT:



C:\Users\admin\Desktop\1WN24CS232\linkedlist.exe

```
1. Insert at Front
2. Insert at End
3. Insert at Position
4. Delete from Front
5. Delete from End
6. Delete from Position
7. Display List
8. Exit
Enter your choice: 1
Enter value to insert at front: 1
Value 1 inserted at the front.
Enter your choice: 2
Enter value to insert at end: 3
Value 3 inserted at the end.
Enter your choice: 7
Linked List: 1 -> 3 -> NULL
Enter your choice: 3
Enter value to insert: 2
Enter position: 2
Value 2 inserted at position 2
Enter your choice: 7
Linked List: 1 -> 2 -> 3 -> NULL
Enter your choice: 2
Enter value to insert at end: 4
Value 4 inserted at the end.
Enter your choice: 4
Element 1 deleted from the front.
Enter your choice: 7
Linked List: 2 -> 3 -> 4 -> NULL
Enter your choice: 5
Element 4 deleted from the end.
Enter your choice: 7
Linked List: 2 -> 3 -> NULL
Enter your choice: 6
Enter position to delete from: 2
Element 3 deleted from position 2
Enter your choice: 7
Linked List: 2 -> NULL
Enter your choice: 8

Process returned 0 (0x0)   execution time : 53.278 s
Press any key to continue.
```

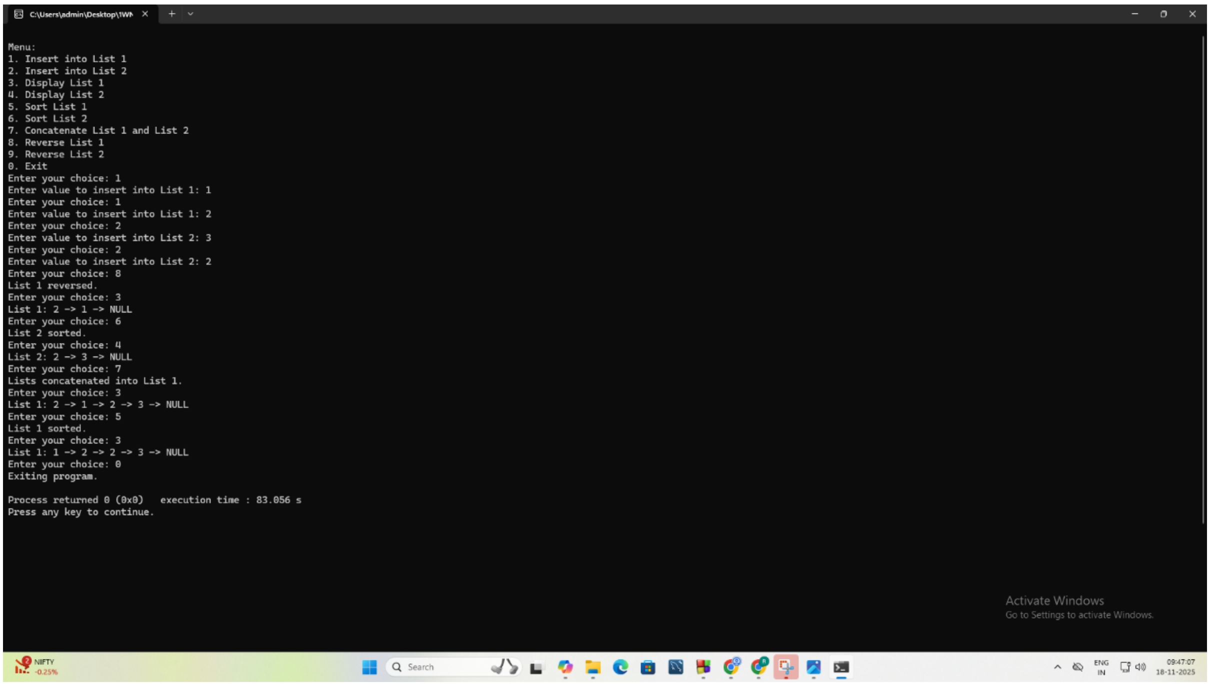
PROGRAM 5:

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 *createNode(int value)
{
   struct node *newNode = (struct node *)malloc(sizeof(struct node));
   newNode->data = value;
   newNode->next = NULL;
   return newNode;
}
void insertEnd(struct node **head, int value)
{
   struct node *newNode = createNode(value);
   if (*head == NULL)
   {
      *head = newNode;
      return;
   }
   struct node *temp = *head;
   while (temp->next != NULL)
   {
      temp = temp->next;
   }
   temp->next = newNode;
}
void display(struct node *head)
{
   if (head == NULL)
   {
      printf("List is empty\n");
      return;
   }
   struct node *temp = head;
   while (temp != NULL)
   {
      printf("%d -> ", temp->data);
      temp = temp->next;
   }
   printf("NULL\n");
}
void sortList(struct node *head)
{
   struct node *i, *j;
   int temp;

   if (head == NULL)
      return;

   for (i = head; i->next != NULL; i = i->next)
```

```
    {
        for (j = i->next; j != NULL; j = j->next)
        {
            if (i->data > j->data)
            {
                temp = i->data;
                i->data = j->data;
                j->data = temp;
            }
        }
    }
}
struct node *reverse(struct node *head)
{
    struct node *prev = NULL, *curr = head, *next = NULL;
    while (curr != NULL)
    {
        next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
struct node *concatenate(struct node *head1, struct node *head2)
{
    if (head1 == NULL)
        return head2;
    if (head2 == NULL)
        return head1;
    struct node *temp = head1;
    while (temp->next != NULL)
    {
        temp = temp->next;
    }
    temp->next = head2;
    return head1;
}
int main()
{
    struct node *list1 = NULL;
    struct node *list2 = NULL;
    int choice, value;
    while (1)
    {
        printf("\n--- MENU ---\n");
        printf("1. Insert into List1\n");
        printf("2. Insert into List2\n");
        printf("3. Display List1\n");
        printf("4. Display List2\n");
        printf("5. Sort List1\n");
        printf("6. Reverse List1\n");
        printf("7. Concatenate List1 + List2\n");
        printf("8. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter value: ");
            scanf("%d", &value);
            insertEnd(&list1, value);
            break;
        case 2:
            printf("Enter value: ");
            scanf("%d", &value);
            insertEnd(&list2, value);
            break;
        case 3:
            printf("List1: ");
            display(list1);
            break;
        case 4:
            printf("List2: ");
            display(list2);
            break;
        case 5:
            sortList(list1);
            printf("List1 sorted.\n");
            break;
        case 6:
            list1 = reverse(list1);
            printf("List1 reversed.\n");
            break;
        case 7:
            list1 = concatenate(list1, list2);
            printf("Concatenated List (List1 + List2):\n");
            display(list1);
            break;
        case 8:
            exit(0);
        default:
            printf("Invalid choice!\n");
        }
    }
}
```

OUTPUT:



```
Menu:
1. Insert into List 1
2. Insert into List 2
3. Display List 1
4. Display List 2
5. Sort List 1
6. Sort List 2
7. Concatenate List 1 and List 2
8. Reverse List 1
9. Reverse List 2
0. Exit
Enter your choice: 1
Enter value to insert into List 1: 1
Enter your choice: 1
Enter value to insert into List 1: 2
Enter your choice: 2
Enter value to insert into List 2: 3
Enter your choice: 2
Enter value to insert into List 2: 2
Enter your choice: 8
List 1 reversed.
Enter your choice: 3
List 1: 2 -> 1 -> NULL
Enter your choice: 6
List 2 sorted.
Enter your choice: 4
List 2: 2 -> 3 -> NULL
Enter your choice: 7
Lists concatenated into List 1.
Enter your choice: 3
List 1: 2 -> 1 -> 2 -> 3 -> NULL
Enter your choice: 5
List 1 sorted.
Enter your choice: 3
List 1: 1 -> 2 -> 2 -> 3 -> NULL
Enter your choice: 0
Exiting program.

Process returned 0 (0x0)   execution time : 83.056 s
Press any key to continue.
```

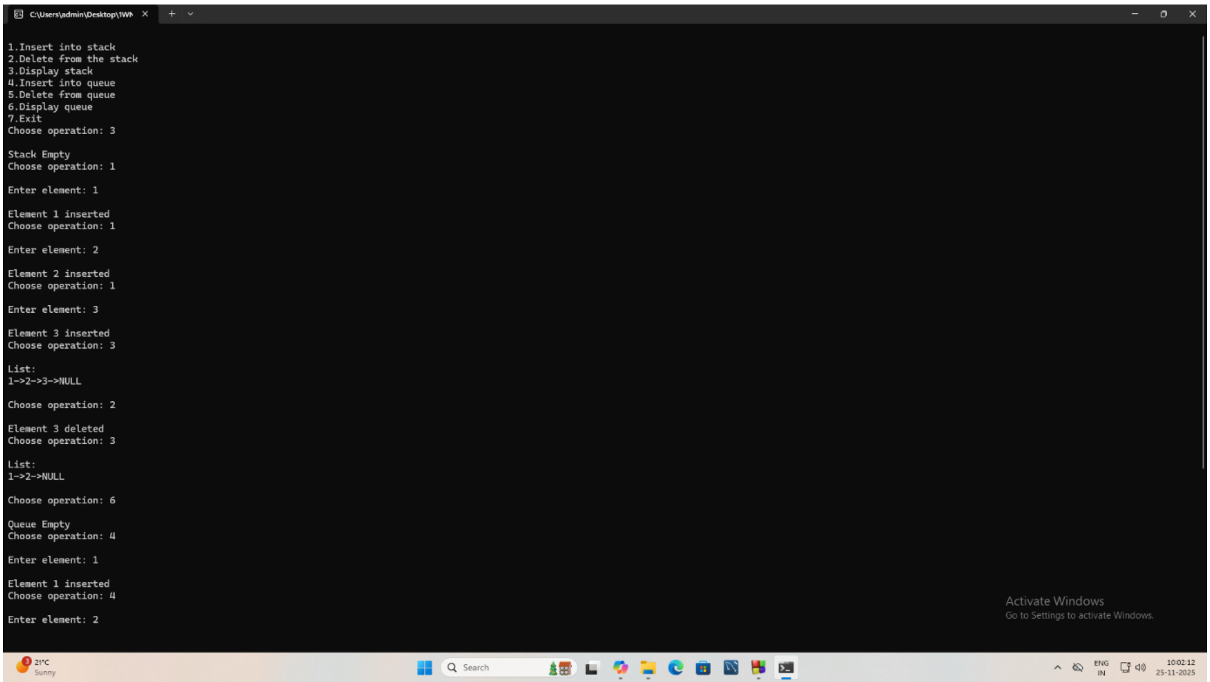
PROGRAM 6:

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

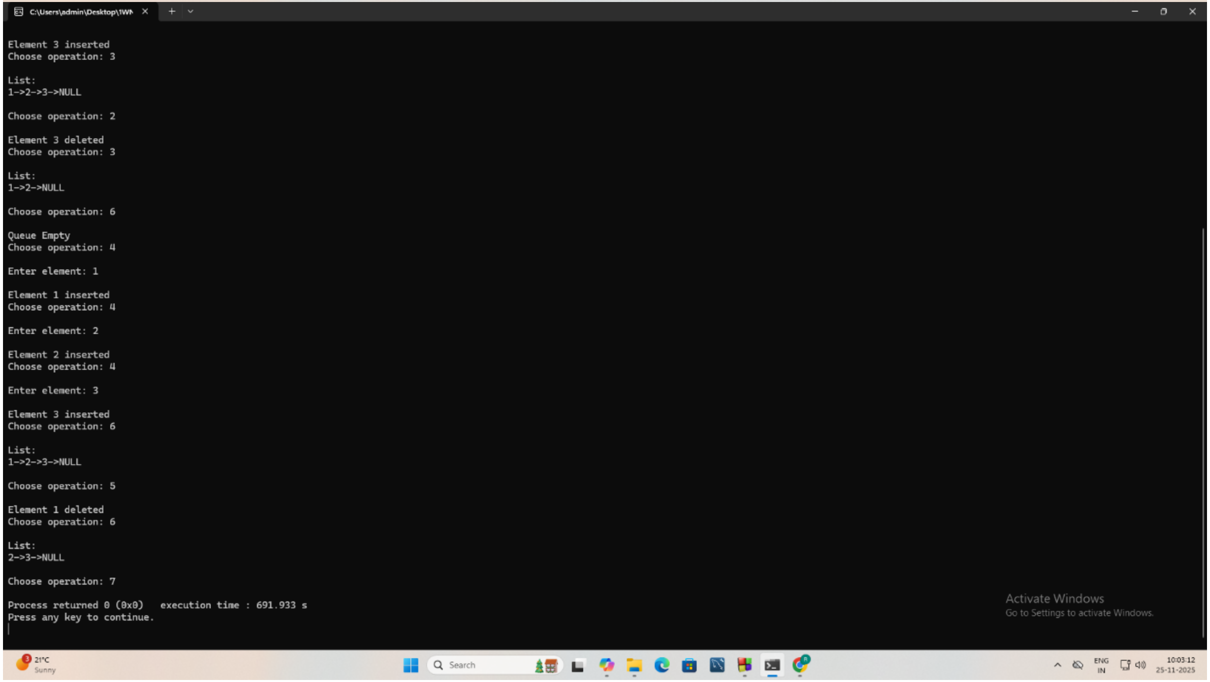
```
#include <stdio.h>
#include <stdlib.h>
struct node {
int data;
struct node *next;
}
;
struct node *top = NULL;
struct node *front = NULL;
struct node *rear = NULL
struct node* createNode(int value) {
    struct node* newNode = (struct node*)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;
    return newNode;
}

/* -------------------- STACK OPERATIONS -------------------- */
void push(int value)
{
    struct node *newNode = createNode(value);
    newNode->next = top;
    top = newNode;
    printf("Pushed %d\n", value);
}
void pop()
{
    if (top == NULL)
    {
        printf("Stack Underflow\n");
        return;
    }
    struct node *temp = top;
    top = top->next;
    printf("Popped %d\n", temp->data);
    free(temp);
}
void displayStack()
{
    if (top == NULL)
    {
        printf("Stack is empty\n");
        return;
    }
    struct node *temp = top;
    printf("Stack: ");
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
```

```
/* -------------------- QUEUE OPERATIONS -------------------- */
void enqueue(int value)
{
    struct node *newNode = createNode(value);
    if (rear == NULL)
    {
        front = rear = newNode;
    }
    else
    {
        rear->next = newNode;
        rear = newNode;
    }
    printf("Enqueued %d\n", value);
}
void dequeue()
{
    if (front == NULL)
    {
        printf("Queue Underflow\n");
        return;
    }
    struct node *temp = front;
    printf("Dequeued %d\n", temp->data);
    front = front->next;
    if (front == NULL)
        rear = NULL;
    free(temp);
}
void displayQueue()
{
    if (front == NULL)
    {
        printf("Queue is empty\n");
        return;
    }
    struct node *temp = front;
    printf("Queue: ");
    while (temp != NULL)
    {
        printf("%d -> ", temp->data);
        temp = temp->next;
    }
    printf("NULL\n");
}
int main()
{
        int choice, value;
        printf("\n--- MENU ---\n");
        printf("1. PUSH (Stack)\n");
        printf("2. POP (Stack)\n");
        printf("3. Display Stack\n");
        printf("4. ENQUEUE (Queue)\n");
        printf("5. DEQUEUE (Queue)\n");
        printf("6. Display Queue\n");
        printf("7. Exit\n");

    while (1)
    {
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice)
        {
        case 1:
            printf("Enter value: ");
            scanf("%d", &value);
            push(value);
            break;
        case 2:
            pop();
            break;
        case 3:
            displayStack();
            break;
        case 4:
            printf("Enter value: ");
            scanf("%d", &value);
            enqueue(value);
            break;
        case 5:
            dequeue();
            break;
        case 6:
            displayQueue();
            break;
        case 7:
            exit(0);
        default:
            printf("Invalid choice! Try again.\n");
        }
    }
}
```

OUTPUT:


C:\Users\admin\Desktop\IWN
Activate Windows
Go to Settings to activate Windows.
21°C
Sunny
Search
ENG
IN
10:02:12
25-11-2025

```
1.Insert into stack
2.Delete from the stack
3.Display stack
4.Insert into queue
5.Delete from queue
6.Display queue
7.Exit
Choose operation: 3

Stack Empty
Choose operation: 1

Enter element: 1

Element 1 inserted
Choose operation: 1

Enter element: 2

Element 2 inserted
Choose operation: 1

Enter element: 3

Element 3 inserted
Choose operation: 3

List:
1->2->3->NULL

Choose operation: 2

Element 3 deleted
Choose operation: 3

List:
1->2->NULL

Choose operation: 6

Queue Empty
Choose operation: 4

Enter element: 1

Element 1 inserted
Choose operation: 4

Enter element: 2
```


C:\Users\admin\Desktop\IWN
Activate Windows
Go to Settings to activate Windows.
21°C
Sunny
Search
ENG
IN
10:03:12
25-11-2025

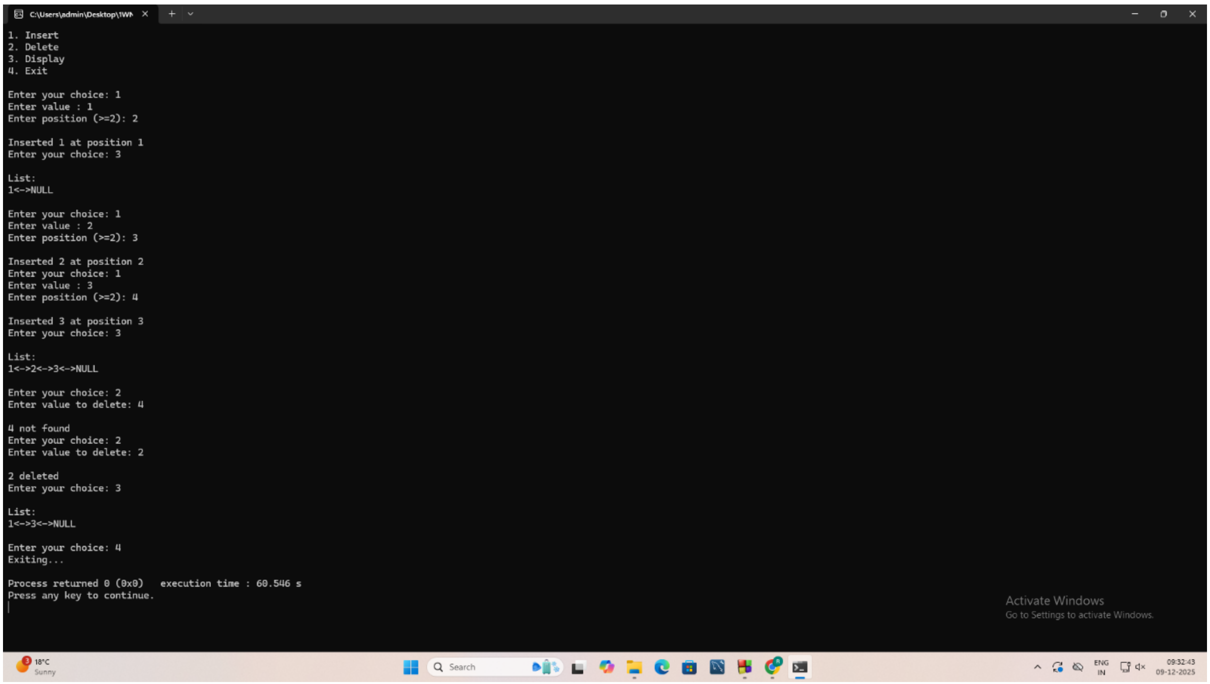
```
Element 3 inserted
Choose operation: 3

List:
1->2->3->NULL

Choose operation: 2

Element 3 deleted
Choose operation: 3

List:
1->2->NULL

Choose operation: 6

Queue Empty
Choose operation: 4

Enter element: 1

Element 1 inserted
Choose operation: 4

Enter element: 2

Element 2 inserted
Choose operation: 4

Enter element: 3

Element 3 inserted
Choose operation: 6

List:
1->2->3->NULL

Choose operation: 5

Element 1 deleted
Choose operation: 6

List:
2->3->NULL

Choose operation: 7

Process returned 0 (0x0)   execution time : 691.933 s
Press any key to continue.
```

PROGRAM 7:
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 {
    int data;
    struct node *prev, *next;
};
struct node *head = NULL;
void insertEnd(int value)
{
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = NULL;
    newNode->prev = NULL;
    if (head == NULL)
    {
        head = newNode;
        return;
    }
    struct node *temp = head;
    while (temp->next != NULL)
        temp = temp->next;
    temp->next = newNode;
    newNode->prev = temp;
}
void insertLeft(int target, int value)
{
    struct node *temp = head;
    while (temp != NULL && temp->data != target)
        temp = temp->next;
    if (temp == NULL)
    {
        printf("Node %d not found!\n", target);
        return;
    }
    struct node *newNode = (struct node *)malloc(sizeof(struct node));
    newNode->data = value;
    newNode->next = temp;
    newNode->prev = temp->prev;
    if (temp->prev != NULL)
        temp->prev->next = newNode;
    else
        head = newNode;
    temp->prev = newNode;
    printf("Inserted %d to the left of %d\n", value, target);
}
void deleteValue(int value)
{
    struct node *temp = head;
    while (temp != NULL && temp->data != value)
```

```
        temp = temp->next;
    if (temp == NULL)
    {
        printf("Value %d not found!\n", value);
        return;
    }
    if (temp->prev != NULL)
        temp->prev->next = temp->next;
    else
        head = temp->next;
    if (temp->next != NULL)
        temp->next->prev = temp->prev;
    printf("Deleted %d\n", temp->data);
    free(temp);
}
void display()
{
    struct node *temp = head;
    if (head == NULL)
    {
        printf("List is empty!\n");
        return;
    }
    printf("List: ");
    while (temp != NULL)
    {
        printf("%d ", temp->data);
        temp = temp->next;
    }
    printf("\n");
}
int main()
{
    int choice, val, target;
    while (1)
    {
        printf("\n--- Doubly Linked List Menu ---\n");
        printf("1. Create/Insert at End\n");
        printf("2. Insert Left of Node\n");
        printf("3. Delete Specific Value\n");
        printf("4. Display List\n");
        printf("5. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);
        switch (choice)
        {
        case 1:
            printf("Enter value: ");
            scanf("%d", &val);
            insertEnd(val);
            break;
        case 2:
            printf("Enter target node value: ");
            scanf("%d", &target);
            printf("Enter new value: ");
            scanf("%d", &val);
            insertLeft(target, val);
            break;
        case 3:
            printf("Enter value to delete: ");
            scanf("%d", &val);
            deleteValue(val);
            break;
        case 4:
            display();
            break;
        case 5:
            return 0;
        default:
            printf("Invalid choice!\n");
        }
    }
```

OUTPUT:



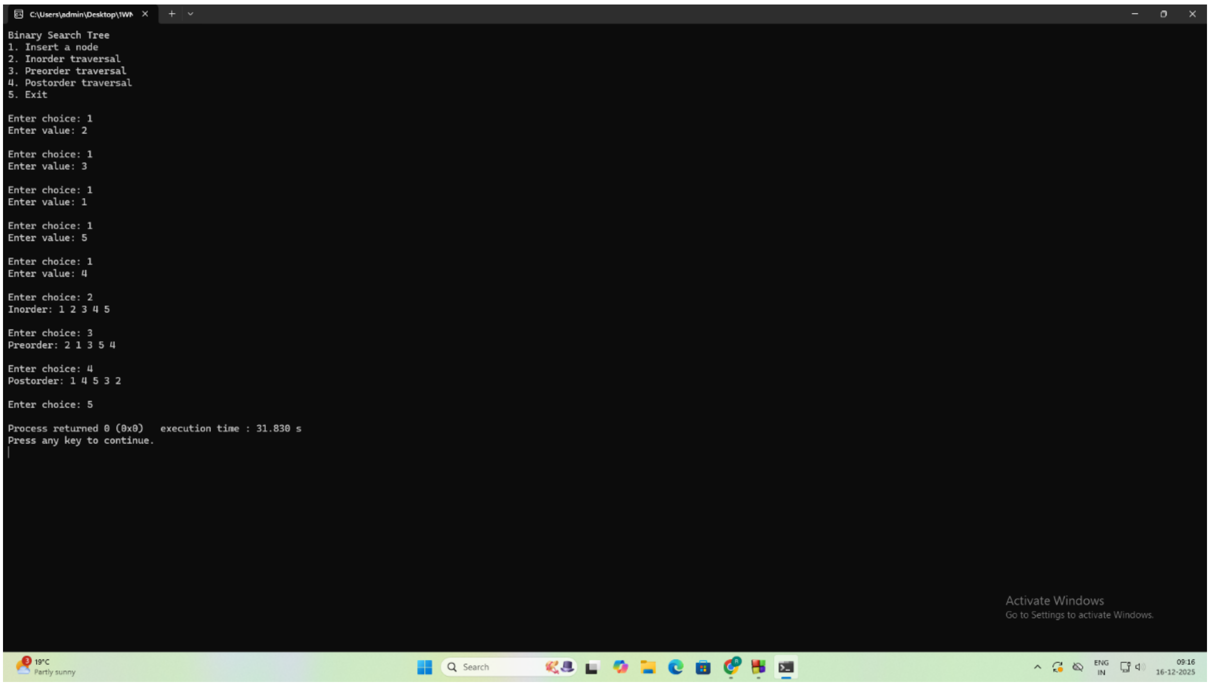
C:\Users\admin\Desktop\1WM
Activate Windows
Go to Settings to activate Windows.
18°C
Sunny
Search
ENG
IN
09:32:43
09-12-2025

```
1. Insert
2. Delete
3. Display
4. Exit

Enter your choice: 1
Enter value : 1
Enter position (>=2): 2

Inserted 1 at position 1
Enter your choice: 3

List:
1<->NULL

Enter your choice: 1
Enter value : 2
Enter position (>=2): 3

Inserted 2 at position 2
Enter your choice: 1
Enter value : 3
Enter position (>=2): 4

Inserted 3 at position 3
Enter your choice: 3

List:
1<->2<->3<->NULL

Enter your choice: 2
Enter value to delete: 4

4 not found
Enter your choice: 2
Enter value to delete: 2

2 deleted
Enter your choice: 3

List:
1<->3<->NULL

Enter your choice: 4
Exiting...

Process returned 0 (0x0)   execution time : 60.546 s
Press any key to continue.
```

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
c) To display the elements in the tree.

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node {
    int data;
    struct node *left;
    struct node *right;
}node;
node *create(int value)
{
    node *newnode = (node *)malloc(sizeof(node));
    newnode->data = value;
    newnode->left = NULL;
    newnode->right = NULL;
}
struct node *insert(node *root, int value)
{
    if (root == NULL)
        return create(value);
    if (value < root->data)
        root->left = insert(root->left, value);
    else if (value > root->data)
        root->right = insert(root->right, value);
    return root;
}
struct node *find_min(node *ptr)
{
    node *current = ptr;
    while (current && current->left != NULL)
        current = current->left;
    return current;
}
node *delete(node *root, int key)
{
    if (root == NULL)
        return root;
    if (key < root->data)
        root->left = delete(root->left, key);
    else if (key > root->data)
        root->right = delete(root->right, key);
    else
    {
        if (root->left == NULL)
        {
            node *temp = root->right;
            free(root);
            return temp;
        }
        else if (root->right == NULL)
        {
            node *temp = root->left;
            free(root);
            return temp;
        }

        node *temp = find_min(root->right);
        root->data = temp->data;
        root->right = delete(root->right, temp->data);
    }
    return root;
}
void inorder_trav(node *root)
{
    if (root != NULL)
    {
        inorder_trav(root->left);
        printf("%d -> ", root->data);
        inorder_trav(root->right);
    }
}
void preorder_trav(node *root)
{
    if (root != NULL)
    {
        printf("%d -> ", root->data);
        preorder_trav(root->left);
        preorder_trav(root->right);
    }
}
void postorder_trav(node *root)
{
    if (root != NULL)
    {
        postorder_trav(root->left);
        postorder_trav(root->right);
        printf("%d -> ", root->data);
    }
}
int main()
{
    node *root = NULL;
    int choice, value;
    while (1)
    {
        printf("\n--- Binary Search Tree Menu ---\n");
        printf("1. Insert a node\n");
        printf("2. Delete a node\n");
        printf("3. Inorder Traversal\n");
        printf("4. Preorder Traversal\n");
        printf("5. Postorder Traversal\n");
        printf("6. Exit\n");
        printf("Enter your choice: ");
        if (scanf("%d", &choice) != 1)
        {
            printf("Invalid input. Exiting.\n");
            break;
        }
        switch (choice)
        {
        case 1:
            printf("Enter value to insert: ");
            scanf("%d", &value);
            root = insert(root, value);
            break;
        case 2:
            printf("Enter value to delete: ");
            scanf("%d", &value);
            root = delete(root, value);
            break;
        case 3:
            printf("Inorder Traversal: ");
            inorder_trav(root);
            printf("NULL\n");
            break;
        case 4:
            printf("Preorder Traversal: ");
            preorder_trav(root);
            printf("NULL\n");
            break;
        case 5:
            printf("Postorder Traversal: ");
            postorder_trav(root);
            printf("NULL\n");
            break;
        case 6:
            printf("Exiting...\n");
            return 0;
        default:
            printf("Invalid choice. Please try again.\n");
        }
    }
    return 0;
}
```

OUTPUT:



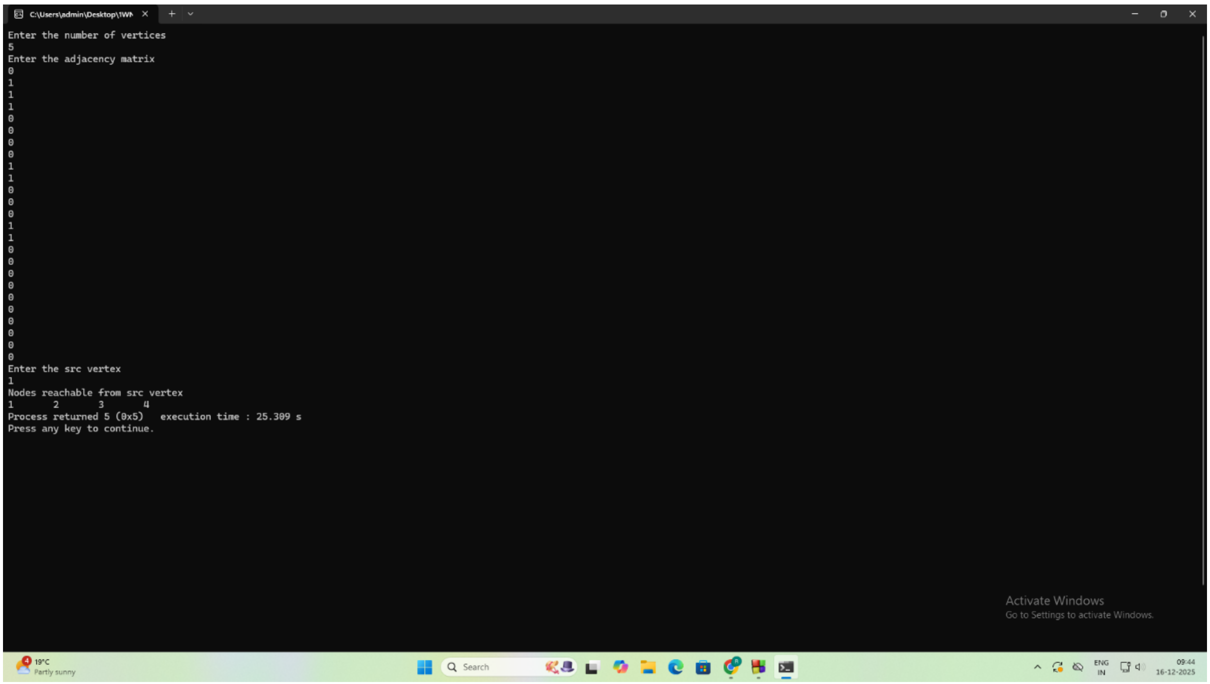
```
Binary Search Tree
1. Insert a node
2. Inorder traversal
3. Preorder traversal
4. Postorder traversal
5. Exit

Enter choice: 1
Enter value: 2

Enter choice: 1
Enter value: 3

Enter choice: 1
Enter value: 1

Enter choice: 1
Enter value: 5

Enter choice: 1
Enter value: 4

Enter choice: 2
Inorder: 1 2 3 4 5

Enter choice: 3
Preorder: 2 1 3 5 4

Enter choice: 4
Postorder: 1 4 5 3 2

Enter choice: 5

Process returned 0 (0x0)   execution time : 31.830 s
Press any key to continue.
```

PROGRAM 9:

Write a program to traverse a graph using BFS method.

```
#include <stdio.h>
void bfs(int);
int a[10][10], vis[10], n;
void main()
{
    int i, j, src;
    printf("enter the number of vertices\n");
    scanf("%d", &n);
    printf("enter the adjacency matrix\n");
    for (i = 1; i <= n; i++)
    {
        for (j = 1; j <= n; j++)
        {
            scanf("%d", &a[i][j]);
        }
        vis[i] = 0;
    }
    printf("enter the src vertex\n");
    scanf("%d", &src);
    printf("nodes reachable from src vertex\n");
    bfs(src);
}
void bfs(int v)
{
    int q[10], f = 1, r = 1, u, i;
    q[r] = v;
    vis[v] = 1;
    while (f <= r)
    {
        u = q[f];
        printf("%d", u);
        for (i = 1; i <= n; i++)
        {
            if (a[v][i] == 1 && vis[i] == 0)
            {
                vis[i] = 1;
                r = r + 1;
                q[r] = i;
            }
        }
        f = f + 1;
    }
}
```

OUTPUT:



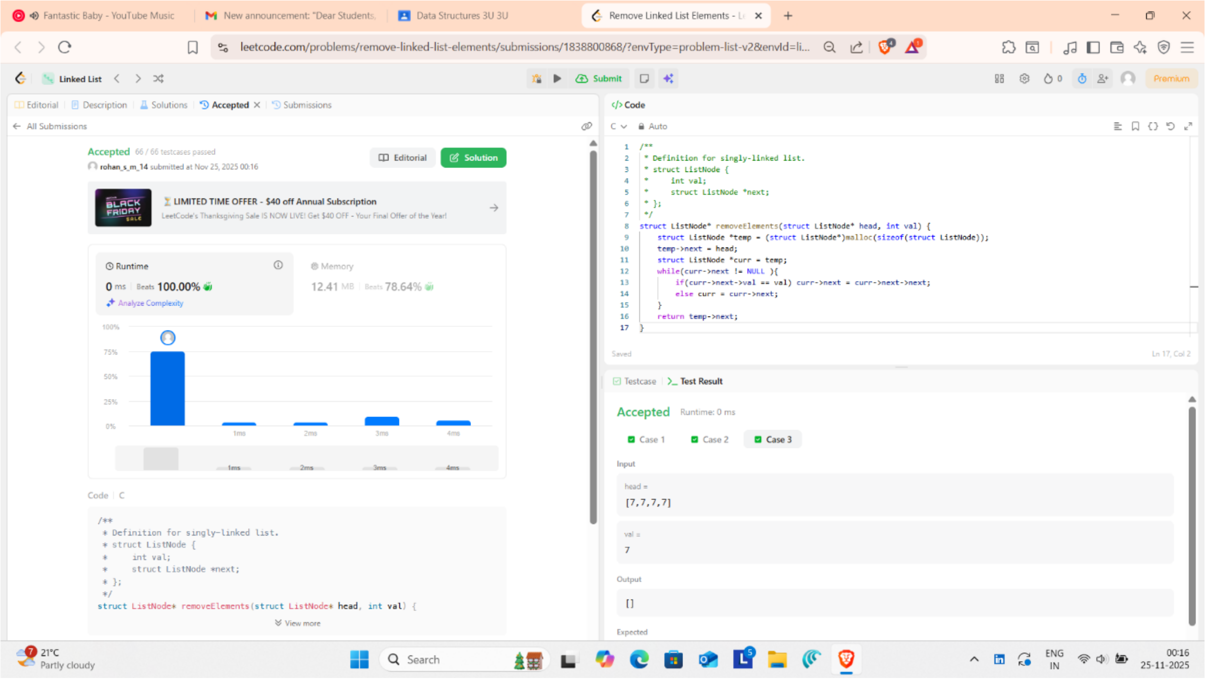
C:\Users\admin\Desktop\1WN
Activate Windows
Go to Settings to activate Windows.
19°C
Partly sunny
Search
ENG
IN
09:44
16-12-2025

```
Enter the number of vertices
5
Enter the adjacency matrix
0
1
1
1
0
0
0
0
1
1
0
0
0
1
1
0
0
0
0
0
0
0
0
0
0
Enter the src vertex
1
Nodes reachable from src vertex
1       2       3       4
Process returned 5 (0x5)   execution time : 25.309 s
Press any key to continue.
```

LEETCODE PROGRAMS

1.deletion of given element

https://leetcode.com/problems/remove-linked-list-elements/description/?envType=problem-list-v2&envId=linked-list



Accepted 66 / 66 testcases passed

rohan_s_m_14 submitted at Nov 25, 2025 00:16

Runtime 0 ms | Beats 100.00%

Memory 12.41 MB | Beats 78.64%

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* removeElements(struct ListNode* head, int val) {
    struct ListNode *temp = (struct ListNode*)malloc(sizeof(struct ListNode));
    temp->next = head;
    struct ListNode *curr = temp;
    while(curr->next != NULL ){
        if(curr->next->val == val) curr->next = curr->next->next;
        else curr = curr->next;
    }
    return temp->next;
}
```

Accepted Runtime: 0 ms

Input

head =

[7,7,7,7]

val =

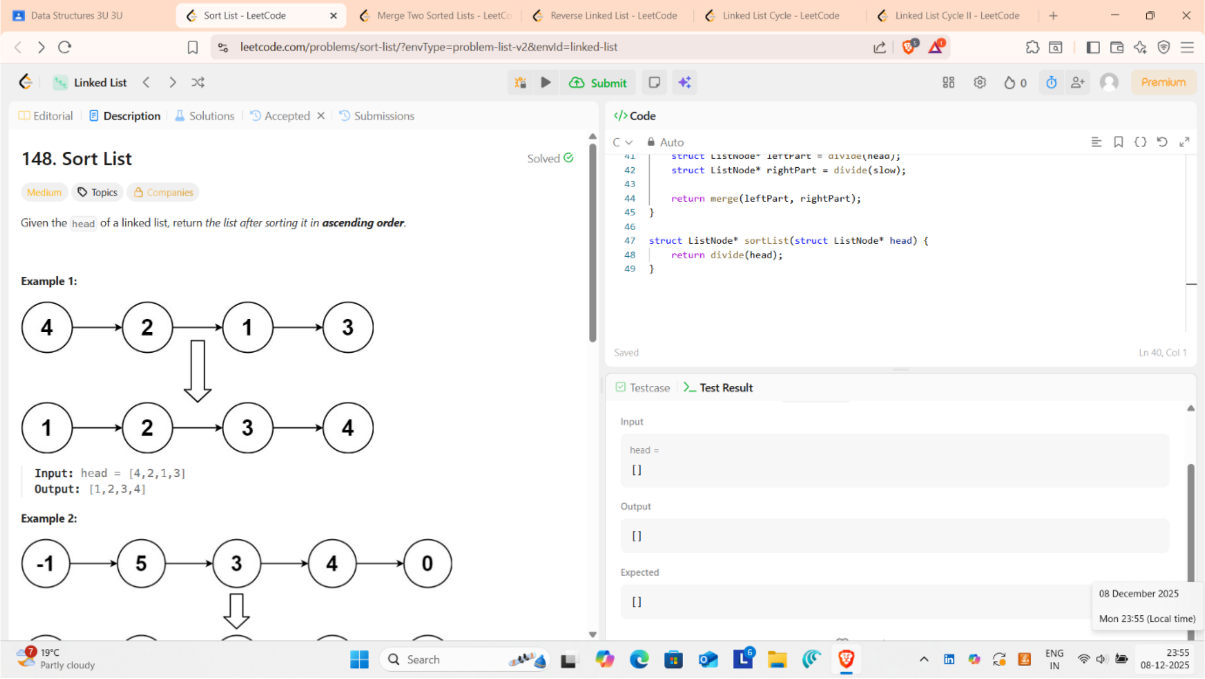
7

Output

[]

2.sortion

https://leetcode.com/problems/sort-list/description/?envType=problem-list-v2&envId=linked-list



148. Sort List

Solved

Medium

Given the head of a linked list, return *the list after sorting it in* ***ascending order***.

Example 1:

```
Input: head = [4,2,1,3]
Output: [1,2,3,4]
```

Example 2:

```c
    struct ListNode* leftPart = divide(head);
    struct ListNode* rightPart = divide(slow);

    return merge(leftPart, rightPart);
}

struct ListNode* sortList(struct ListNode* head) {
    return divide(head);
}
```

Input

head =

[]

Output

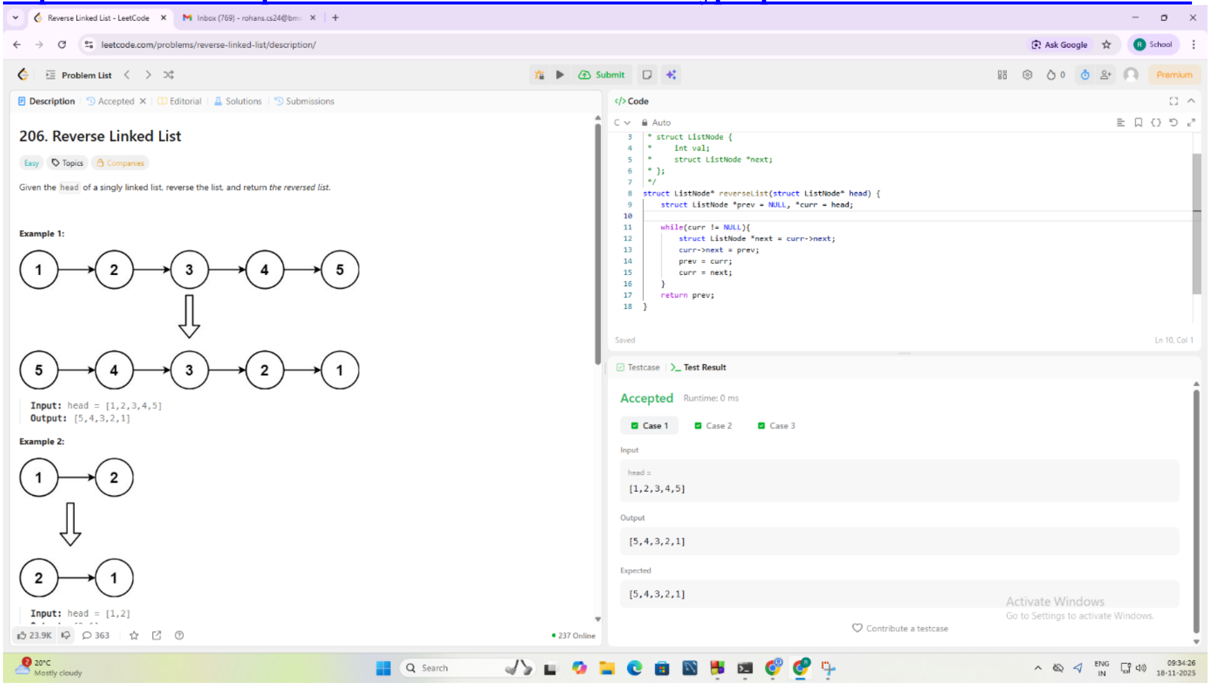
[]

Expected

[]

3.reversing

https://leetcode.com/problems/reverse-linked-list/?envType=problem-list-v2&envId=linked-list



206. Reverse Linked List

Easy | Topics | Companies

Given the `head` of a singly linked list, reverse the list, and return *the reversed list.*

Example 1:

```
Input: head = [1,2,3,4,5]
Output: [5,4,3,2,1]
```

Example 2:

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

```c
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
struct ListNode* reverseList(struct ListNode* head) {
    struct ListNode *prev = NULL, *curr = head;

    while(curr != NULL){
        struct ListNode *next = curr->next;
        curr->next = prev;
        prev = curr;
        curr = next;
    }
    return prev;
}
```

Accepted Runtime: 0 ms

Case 1 Case 2 Case 3

Input

```
head =
[1,2,3,4,5]
```

Output

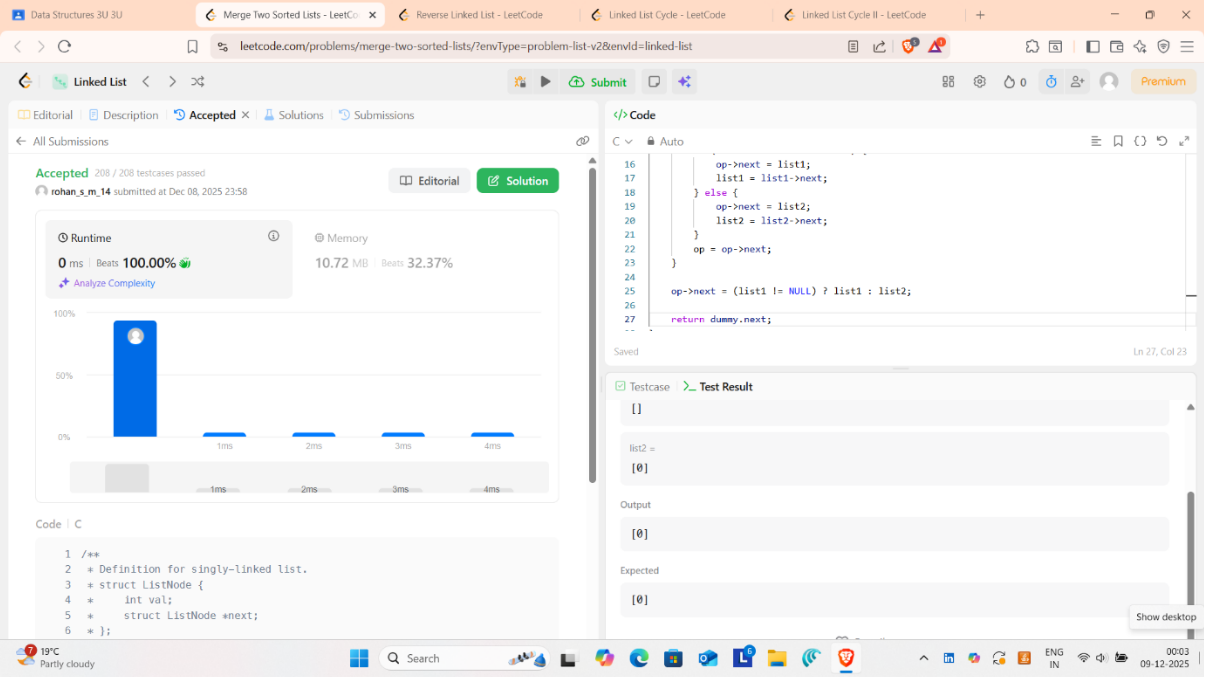
```
[5,4,3,2,1]
```

Expected

```
[5,4,3,2,1]
```

4.concat or merging

https://leetcode.com/problems/merge-two-sorted-lists/description/?envType=problem-list-v2&envId=linked-list



Accepted 208 / 208 testcases passed

rohan_s_m_14 submitted at Dec 08, 2025 23:58

Runtime 0 ms | Beats 100.00%

Memory 10.72 MB | Beats 32.37%

Code | C

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

```c
                op->next = list1;
                list1 = list1->next;
            } else {
                op->next = list2;
                list2 = list2->next;
            }
            op = op->next;
        }

        op->next = (list1 != NULL) ? list1 : list2;

        return dummy.next;
```

```
[]
```

list2 =

```
[0]
```

Output

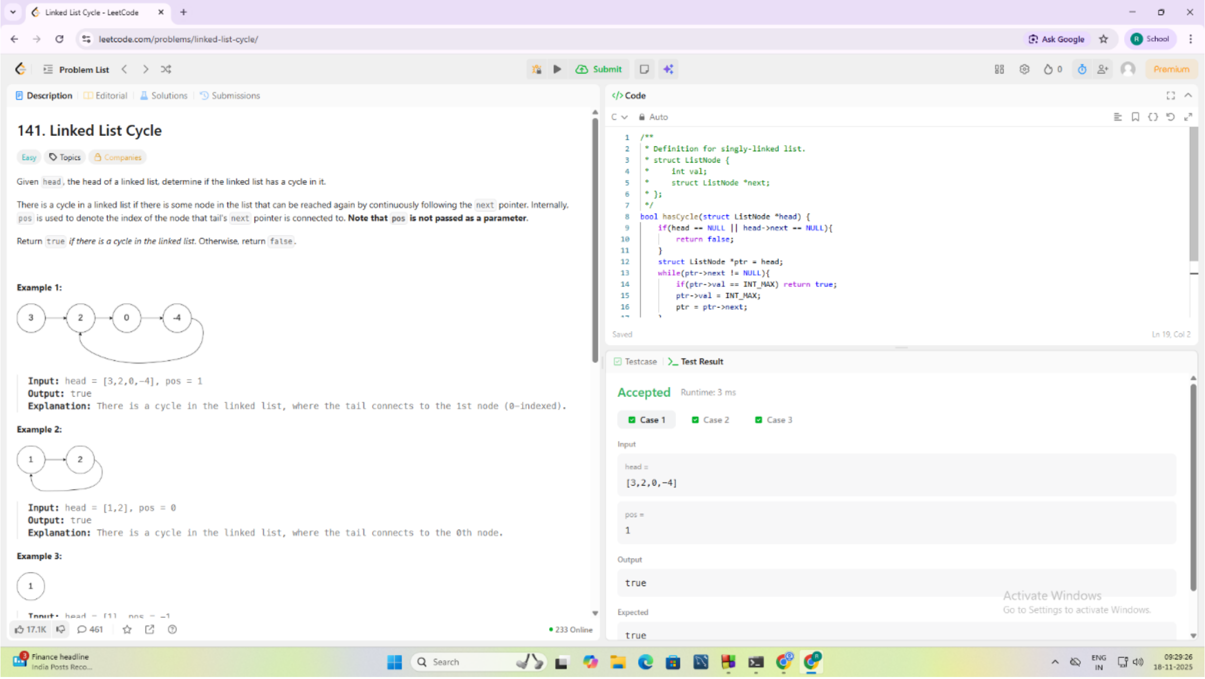
```
[0]
```

Expected

```
[0]
```

5.list cycle 1

https://leetcode.com/problems/linked-list-cycle/description/?envType=problem-list-v2&envId=linked-list


Linked List Cycle - LeetCode
leetcode.com/problems/linked-list-cycle/
Problem List
Submit
Description
Editorial
Solutions
Submissions
Code
Auto

141. Linked List Cycle

Easy · Topics · Companies

Given `head`, the head of a linked list, determine if the linked list has a cycle in it.

There is a cycle in a linked list if there is some node in the list that can be reached again by continuously following the `next` pointer. Internally, `pos` is used to denote the index of the node that tail's `next` pointer is connected to. **Note that `pos` is not passed as a parameter.**

Return `true` *if there is a cycle in the linked list*. Otherwise, return `false`.

Example 1:

```
Input: head = [3,2,0,-4], pos = 1
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 1st node (0-indexed).
```

Example 2:

```
Input: head = [1,2], pos = 0
Output: true
Explanation: There is a cycle in the linked list, where the tail connects to the 0th node.
```

Example 3:

```
Input: head = [1], pos = -1
```

```c
/**
 * Definition for singly-linked list.
 * struct ListNode {
 *     int val;
 *     struct ListNode *next;
 * };
 */
bool hasCycle(struct ListNode *head) {
    if(head == NULL || head->next == NULL){
        return false;
    }
    struct ListNode *ptr = head;
    while(ptr->next != NULL){
        if(ptr->val == INT_MAX) return true;
        ptr->val = INT_MAX;
        ptr = ptr->next;
```

Saved · Ln 19, Col 2

Testcase | Test Result

Accepted Runtime: 3 ms

Case 1 · Case 2 · Case 3

Input

```
head =
[3,2,0,-4]
pos =
1
```

Output

```
true
```

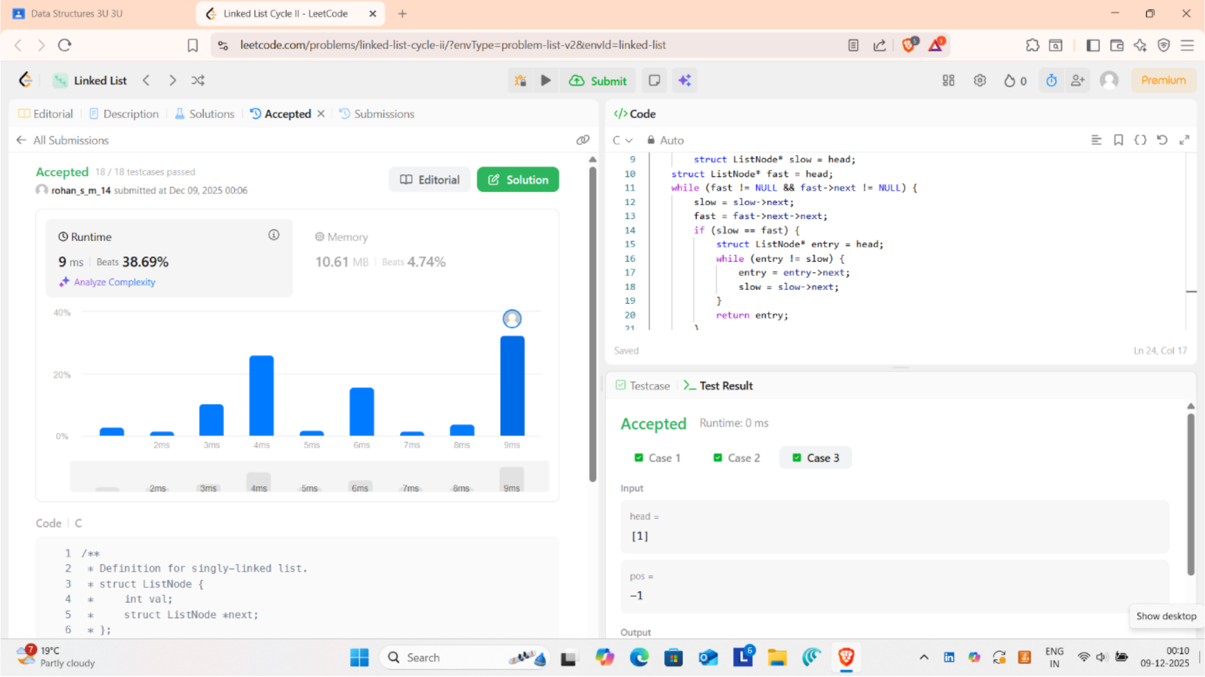
Expected

```
true
```

17.1K · 461 · 233 Online

6.list cycle 2

https://leetcode.com/problems/linked-list-cycle-ii/description/?envType=problem-list-v2&envId=linked-list



Accepted 18 / 18 testcases passed

rohan_s_m_14 submitted at Dec 09, 2025 00:06

Runtime: 9 ms Beats 38.69%

Analyze Complexity

Memory: 10.61 MB Beats 4.74%

Code | C

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

```c
        struct ListNode* slow = head;
    struct ListNode* fast = head;
    while (fast != NULL && fast->next != NULL) {
        slow = slow->next;
        fast = fast->next->next;
        if (slow == fast) {
            struct ListNode* entry = head;
            while (entry != slow) {
                entry = entry->next;
                slow = slow->next;
            }
            return entry;
```

Saved · Ln 24, Col 17

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2 · Case 3

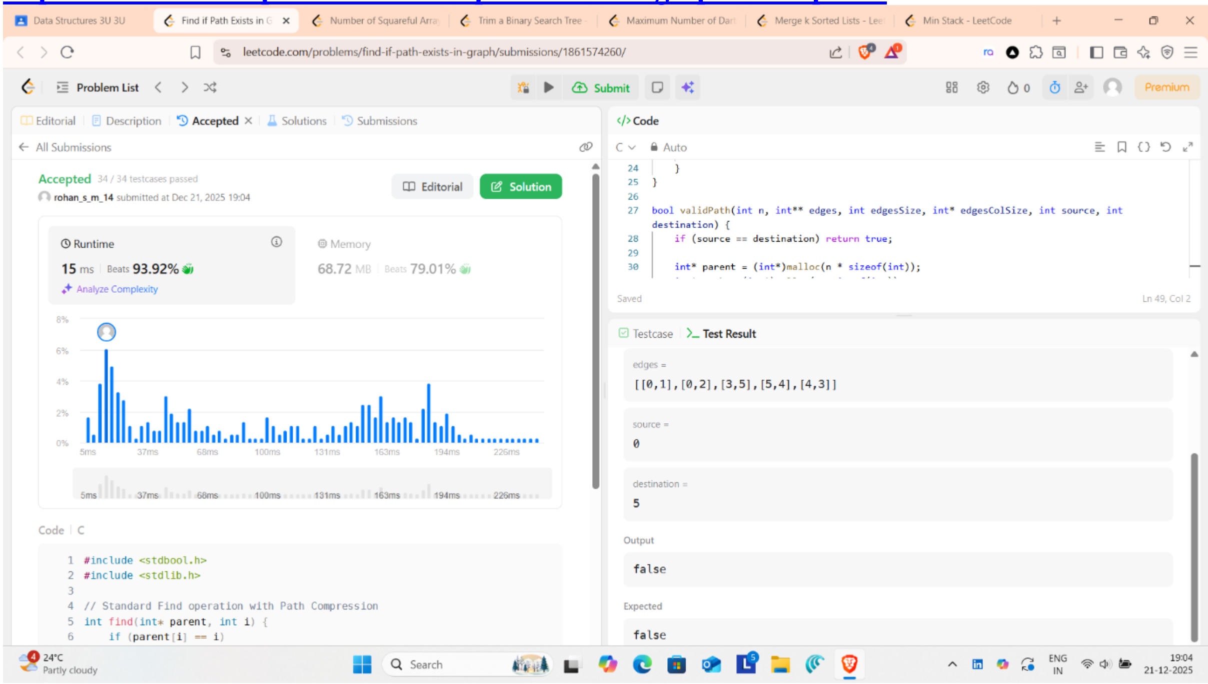
Input

```
head =
[1]
pos =
-1
```

Output

7. Find if path exists in the graph

https://leetcode.com/problems/find-if-path-exists-in-graph/description/



leetcode.com/problems/find-if-path-exists-in-graph/submissions/1861574260/

Accepted 34 / 34 testcases passed

rohan_s_m_14 submitted at Dec 21, 2025 19:04

Runtime 15 ms | Beats 93.92%

Memory 68.72 MB | Beats 79.01%

Code | C

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

// Standard Find operation with Path Compression
int find(int* parent, int i) {
    if (parent[i] == i)
```

Code

C Auto

```
    }
}

bool validPath(int n, int** edges, int edgesSize, int* edgesColSize, int source, int
destination) {
    if (source == destination) return true;

    int* parent = (int*)malloc(n * sizeof(int));
```

Saved Ln 49, Col 2

Testcase | Test Result

edges =

```
[[0,1],[0,2],[3,5],[5,4],[4,3]]
```

source =

```
0
```

destination =

```
5
```

Output

```
false
```

Expected

```
false
```

THANK YOU