VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

DATA STRUCTURES

Submitted by

PALLE PADMAVATHI (1BM21CS125)

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
Oct 2022-Feb 2023

B. M. S. College of Engineering,

Bull Temple Road, Bangalore 560019

(Affiliated To Visvesvaraya Technological University, Belgaum)

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "**DATA STRUCTURES**" carried out by **PALLE PADMAVATHI(1BM21CS125),** 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 2022-23. The Lab report has been approved as it satisfies the academic requirements in respect of Data structures Lab **- (22CS3PCDST)**work prescribed for the said degree.

Name of the Lab-Incharge : Rajeswari madli
Designation: Assistant professor
Department of CSE
BMSCE, Bengaluru

Dr. JyothiNayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

`

Index Sheet

Course Outcome

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 structure

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.

CODE:

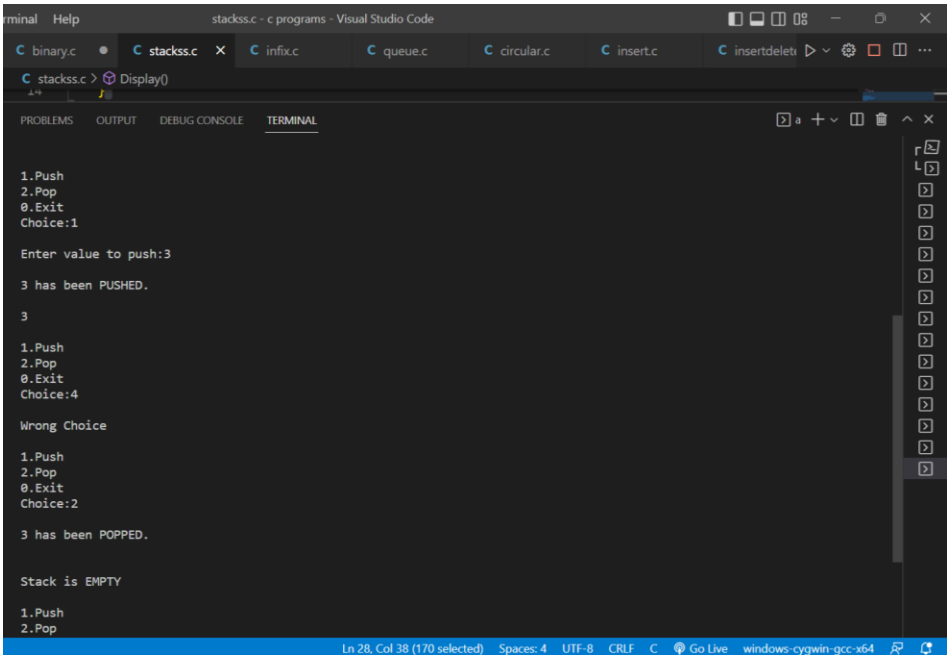
```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5


int top = -1;
int i;
int stk[MAX];


int isEmpty(){
    if(top == -1)
        return 1;
    else
        return 0;
    }



int isFull(){
    if(top == MAX-1)
        return 1;
```

```
    else
        return 0;
    }



void Display(){
    if(isEmpty() == 0){
        for(i=0; i<=top; i++)
                printf("%d ",stk[i]);
        printf("\n");
        }
    else
        printf("\nStack is EMPTY\n");
    }



void Push(){
    int val;
    if(isFull() == 0){
        top++;
        printf("\nEnter value to push:");
        scanf("%d", &val);
        stk[top] = val;
        printf("\n%d has been PUSHED.\n\n", val);
        }
    else
        printf("\nStack is FULL\n");
```

```
    Display();
    }


void Pop(){
    if(isEmpty() == 1)
        printf("\nStack is EMPTY\n");
    else{
        printf("\n%d has been POPPED.\n\n", stk[top]);
        top--;
        }

    Display();
    }


int main(){
    int c=1;
    do{
        printf("\n1.Push \n2.Pop \n0.Exit\nChoice:");
        scanf("%d", &c);
        switch(c){
            case 1:
                Push();
                break;
            case 2:
```

```
                Pop();
                break;
            case 0:
                exit(1);
            default:
                printf("\nWrong Choice\n");
            }
        }while(c != 0);
    return 0;
    }
```

OUTPUT :



LAB PROGRAM2:

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) .

CODE:

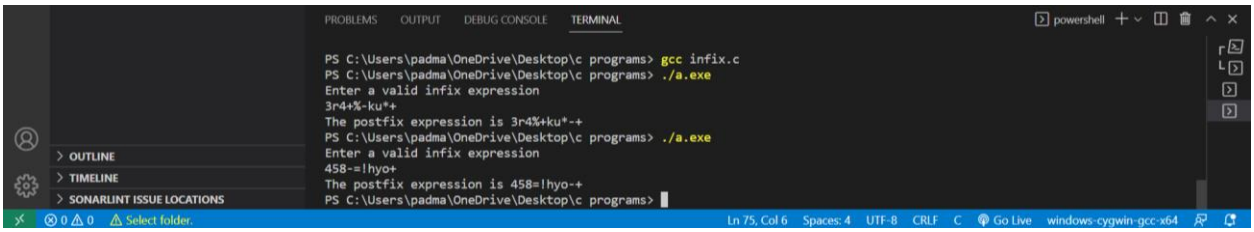
```
#include<stdio.h>
#include<string.h>
int top=-1;
char infix[20],postfix[20],s[20];
void inf_to_post();
int sp(char);
int ip(char);
void push(char);
char pop();
void main()
{
    printf("Enter a valid infix expression\n");
    scanf("%s",infix);
    inf_to_post();
    printf("The postfix expression is %s",postfix);
}
void push(char item)
{
    s[++top]=item;
}
char pop()
```

```
{
    return s[top--];
}
int sp(char item)
{
    switch(item)
    {
        case'+':
        case'-':return 2;
        case'*':
        case'/':return 4;
        case'^':return 5;
        case'(':return 0;
        case'#':return-1;
        default:return 8;
    }
}
int ip(char item)
{
    switch(item)
    {
        case'+':
        case'-':return 1;
        case'*':
        case'/':return 3;
        case'^':return 6;
        case'(':return 9;
```

```
        case')':return 0;
        default:return 7;
    }
}
void inf_to_post()
{
    int i,j=0;
    char symbol;
    push('#');
    for(i=0;i<strlen(infix);i++)
    {
        symbol=infix[i];
        while(sp(s[top])>ip(symbol))
        {
            postfix[j]=pop();
            j++;
        }
        if(sp(s[top])<ip(symbol))
        {
            push(symbol);
        }
        if(sp(s[top])==ip(symbol))
        {
            pop();
        }

    }
```

```
    while(s[top]!='#')
    {
        postfix[j]=pop();
        j++;
    }
    postfix[j]='\0';


}
```

OUTPUT:



LAB PROGRAM 3:

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

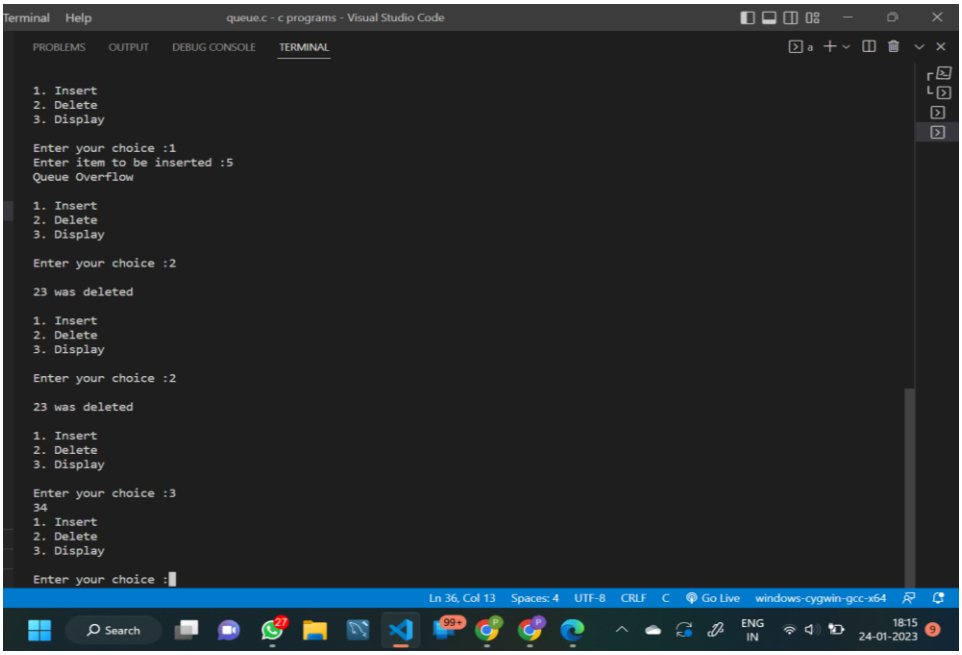
CODE:

```
#include<stdio.h>

#include<stdlib.h>

#define QSIZE 3
```

```c
void insert_rear(int q[],int item, int*r)
{
    if(*r==QSIZE-1)
        printf("Queue Overflow\n");
    else
    {
        (*r)++;
        q[*r]=item;
    }

}
int delete_front(int q[],int *r,int *f)
{
    if(*f>*r)
        printf("Queue underflow\n");
    else
        return q[(*f)++];
}
void display(int q[],int *r,int *f)
{
    int i;
    if(*f>*r)
        printf("Queue is empty\n");
    else
    {
        for(i=*f;i<=*r;i++)
            printf("%d\t",q[i]);
```

```
    }
}
void main()
{
    int q[QSIZE],r=-1, f=0, c,val_del,item;
    while(1)
    {
        printf("\n1. Insert\n2. Delete\n3. Display\n");
        printf("\nEnter your choice :");
        scanf("%d",&c);
        switch(c)
        {
            case 1: printf("Enter item to be inserted :");
                    scanf("%d",&item);
                    insert_rear(q,item,&r);
                    break;
            case 2: val_del=delete_front(q,&r,&f);
                    printf("\n%d was deleted\n",val_del);
                    break;
            case 3: display(q,&r,&f);
                    break;
            default: printf("\nInvalid choice!!!");
                     exit(0);
                     break;
        }
    }
}
```

OUTPUT:



LAB PROGRAM 4:

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

CODE:

```
#include<stdio.h>

# define MAX 5

int cqueue_arr[MAX];

int front = -1;

int rear = -1;

void insert(int item)

{
```

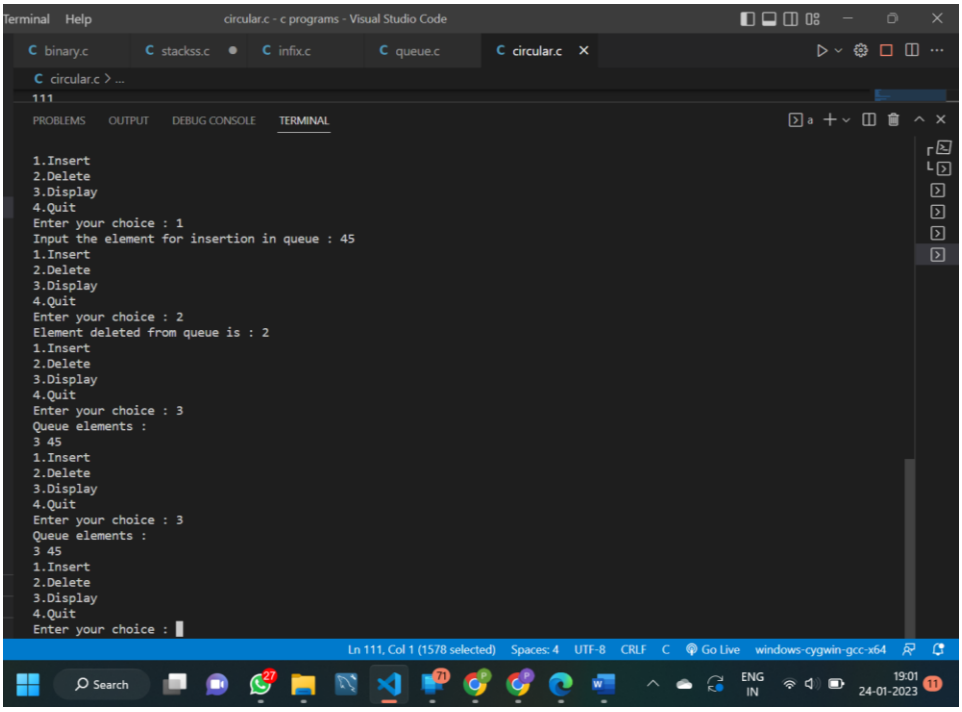
```
if((front == 0 && rear == MAX-1) || (front == rear+1))
{
printf("Queue Overflow \n");
return;
}
if(front == -1)
{
front = 0;
rear = 0;
}
else
{
if(rear == MAX-1)
rear = 0;
else
rear = rear+1;
}
cqueue_arr[rear] = item ;
}
void deletion()
{
if(front == -1)
{
printf("Queue Underflow\n");
return ;
}
printf("Element deleted from queue is : %d\n",cqueue_arr[front]);
```

```
if(front == rear)
{
front = -1;
rear=-1;
}
else
{
if(front == MAX-1)
front = 0;
else
front = front+1;
}
}
void display()
{
int front_pos = front,rear_pos = rear;
if(front == -1)
{
printf("Queue is empty\n");
return;
}
printf("Queue elements :\n");
if( front_pos <= rear_pos )
while(front_pos <= rear_pos)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
```

```
}
else
{
while(front_pos <= MAX-1)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
front_pos = 0;
while(front_pos <= rear_pos)
{
printf("%d ",cqueue_arr[front_pos]);
front_pos++;
}
}
printf("\n");
}
int main()
{
int choice,item;
do
{
printf("1.Insert\n");
printf("2.Delete\n");
printf("3.Display\n");
printf("4.Quit\n");
printf("Enter your choice : ");
```

```
scanf("%d",&choice);
switch(choice)
{
case 1 :
printf("Input the element for insertion in queue : ");
scanf("%d", &item);
insert(item);
break;
case 2 :
deletion();
break;
case 3:
display();
break;
case 4:
break;
default:
printf("Wrong choicen");
}
}while(choice!=4);
return 0;
}
```

OUTPUT:



LAB PROGRAM 5:

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. c) Display the contents of the linked list.

CODE:

```c
#include<stdio.h>

#include<stdlib.h>



typedef struct node{

    int data;

    struct node *next;

}*NODE;
```

```
NODE getnode(){
    NODE p;
    p=(NODE)malloc(sizeof(struct node));
    if(p != NULL)
        return p;
    else{
        printf("No memory allocation.\n");
        exit(0);
    }
}

NODE insertFront(NODE head, int item){
    NODE p;
    p = getnode();
    p->data = item;
    p->next = head;
    head = p;
    return head;
}

NODE insertRear(NODE head, int item){
    NODE p, q;
    q = getnode();
    q->data = item;
    q->next = NULL;

    if(head == NULL)
```

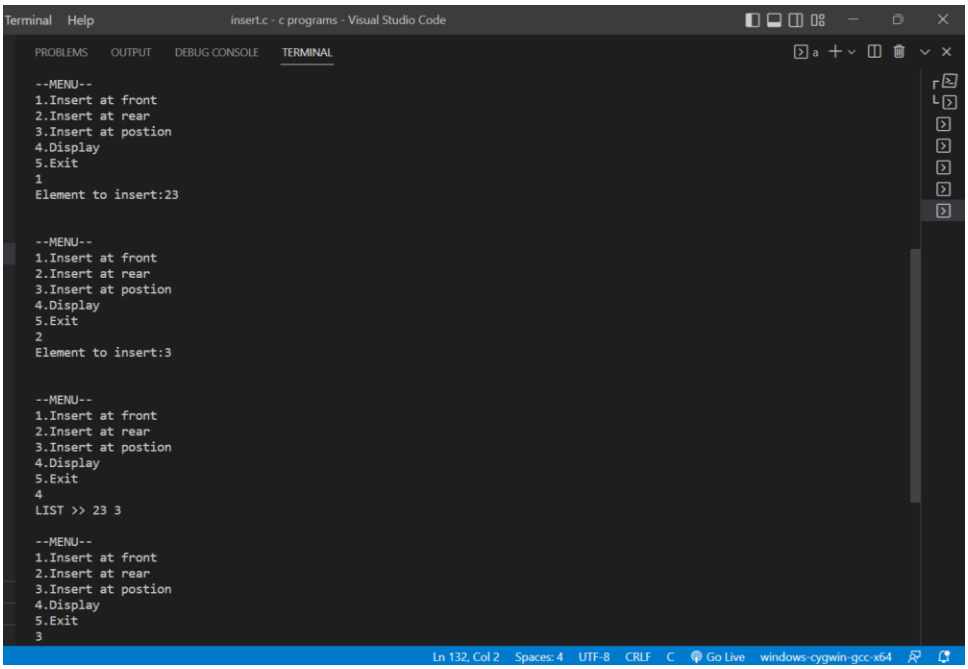
```
        return q;
    p = head;
    while(p->next != NULL)
        p = p->next;
    p->next = q;
    return head;
}

NODE insertPos(NODE head, int item, int pos){
    NODE curr, prev = NULL, newn;
    int count=1;

    newn = getnode();
    newn->data = item;
    newn->next = NULL;

    if(head == NULL){
        if(pos==1)
            return newn;
        else{
            printf("Invalid position\n");
            return 0;
        }
    }

    if(pos == 1){
        newn->next = head;
```

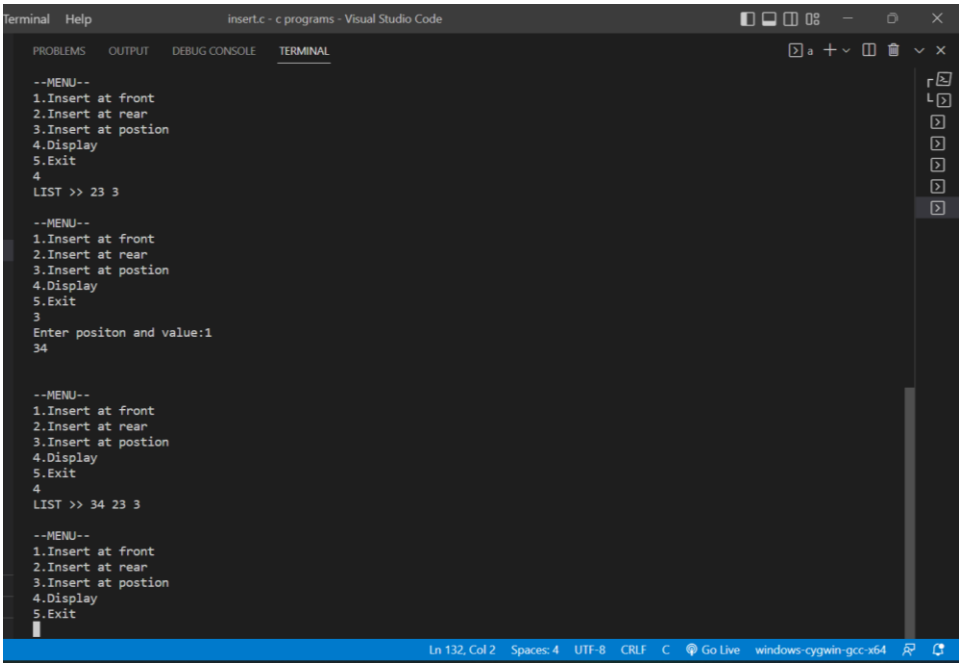
```
        head = newn;
        return head;
    }
    else{
        curr = head;
        while(curr != NULL  &&  count != pos){
            prev = curr;
            curr = curr->next;
            count++;
        }

        if(count == pos){
            prev->next = newn;
            newn->next = curr;
            return head;
        }
        else{
            printf("Invalid position\n");
            return head;
        }
    }
}

void display(NODE head){
    NODE p;
    if(head == NULL){
        printf("List is empty\n");
```

```
        exit(0);
    }
    p = head;
    while(p != NULL){
        printf("%d ", p->data);
        p = p->next;
    }
}


void main()
{
    NODE head = NULL;
    int c, ele, pos, value;


    while(1){
        printf("\n\n--MENU--\n");
        printf("1.Insert at front\n2.Insert at rear\n3.Insert at
postion\n4.Display\n5.Exit\n");
        scanf("%d", &c);
        switch(c){
            case 1:
                printf("Element to insert:");
                scanf("%d", &ele);
                head = insertFront(head, ele);
                break;
            case 2:
                printf("Element to insert:");
```

```
                scanf("%d", &ele);
                head = insertRear(head, ele);
                break;
            case 3:
                printf("Enter positon and value:");
                scanf("%d %d", &pos, &value);
                head = insertPos(head, value, pos);
                break;
            case 4:
                printf("LIST >> ");
                display(head);
                break;
            case 5:
                exit(1);
        }
    }
}
```

OUTPUT:


Terminal Help
insert.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
1
Element to insert:23
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
2
Element to insert:3
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
4
LIST >> 23 3
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
3
Ln 132, Col 2 Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64


Terminal Help
insert.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
4
LIST >> 23 3
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
3
Enter positon and value:1
34
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
4
LIST >> 34 23 3
--MENU--
1.Insert at front
2.Insert at rear
3.Insert at postion
4.Display
5.Exit
Ln 132, Col 2 Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64

LAB PROGRAM6:

WAP to Implement Singly Linked List with following operations a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list.

CODE:

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

typedef struct node{
    int data;
    struct node *next;
}*NODE;

NODE getnode(){
    NODE p;
    p=(NODE)malloc(sizeof(struct node));
    if(p != NULL)
        return p;
    else{
        printf("No memory allocation.\n");
        exit(0);
    }
```

```
}


/*
 *  --- Insert Functions ---
 */


NODE insertFront(NODE head, int item){
    NODE p;
    p = getnode();
    p->data = item;
    p->next = head;
    head = p;
    return head;
}


NODE insertRear(NODE head, int item){
    NODE p, q;
    q = getnode();
    q->data = item;
    q->next = NULL;


    if(head == NULL)
        return q;
    p = head;
    while(p->next != NULL)
        p = p->next;
    p->next = q;
```

```
    return head;
}

NODE insertPos(NODE head, int item, int pos){
    NODE curr, prev = NULL, newn;
    int count=1;

    newn = getnode();
    newn->data = item;
    newn->next = NULL;

    if(head == NULL){
        if(pos == 1)
            return newn;
        else{
            printf("Invalid position\n");
            return 0;
        }
    }

    if(pos == 1){
        newn->next = head;
        head = newn;
        return head;
    }
    else{
        curr = head;
```

```
        while(curr != NULL   &&   count != pos){
            prev = curr;
            curr = curr->next;
            count++;
        }

        if(count == pos){
            prev->next = newn;
            newn->next = curr;
            return head;
        }
        else{
            printf("Invalid position\n");
            return head;
        }
    }
}

/*
 *   --- Delete Functions ---
 */

NODE deleteFront(NODE head){
    NODE p;
    if(head == NULL)
        printf("List is empty");
    else{
```

```
        printf("Deleted %d\n", head->data);
        p = head->next;
        free(head);
        return p;
    }
}

NODE deleteRear(NODE head){
    NODE p,prev;
    p = head;
    if(head == NULL)
        printf("List is empty\n");
    else{
        if(p->next == NULL){
            printf("Deleted %d\n", p->data);
            free(p);
            head = NULL;
        }
        while(p->next != NULL){
            prev = p;
            p = p->next;
        }
        printf("Deleted %d\n", p->data);
        free(p);
        prev->next = NULL;
    }
    return head;
```

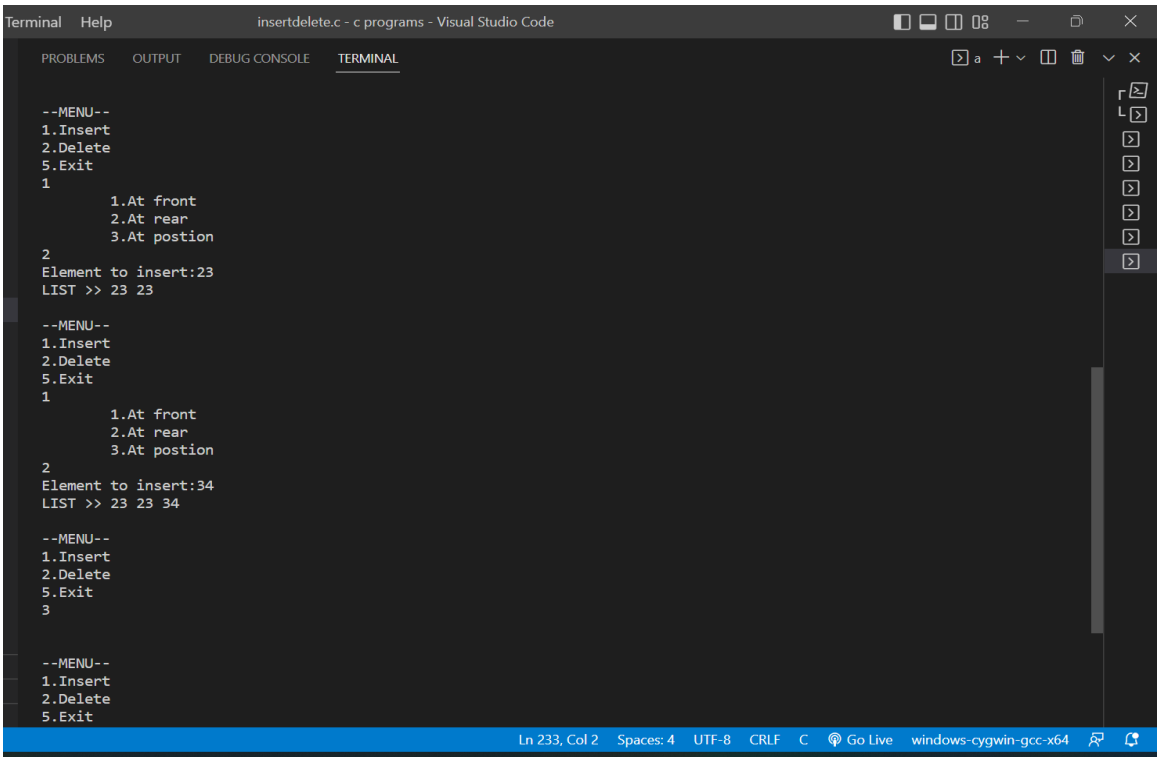
```
}


NODE deletePos(NODE head, int pos){
    NODE p = head;
    if(head == NULL)
        printf("List is empty\n");
    else if(pos == 0){
        printf("Deleted %d\n", head->data);
        NODE temp = head->next;
        free(head);
        head = temp;
    }
    else{
        for(int i=0; i<pos-1; i++){
            p = p->next;
        }
        // now p at element 1 less than given pos
        if(p->next == NULL){
            printf("Invalid Postion\n");
        }
        else{
            NODE temp = p->next;
            printf("Deleted %d\n", temp->data);
            p->next = (p->next)->next;
            free(temp);
        }
    }
```

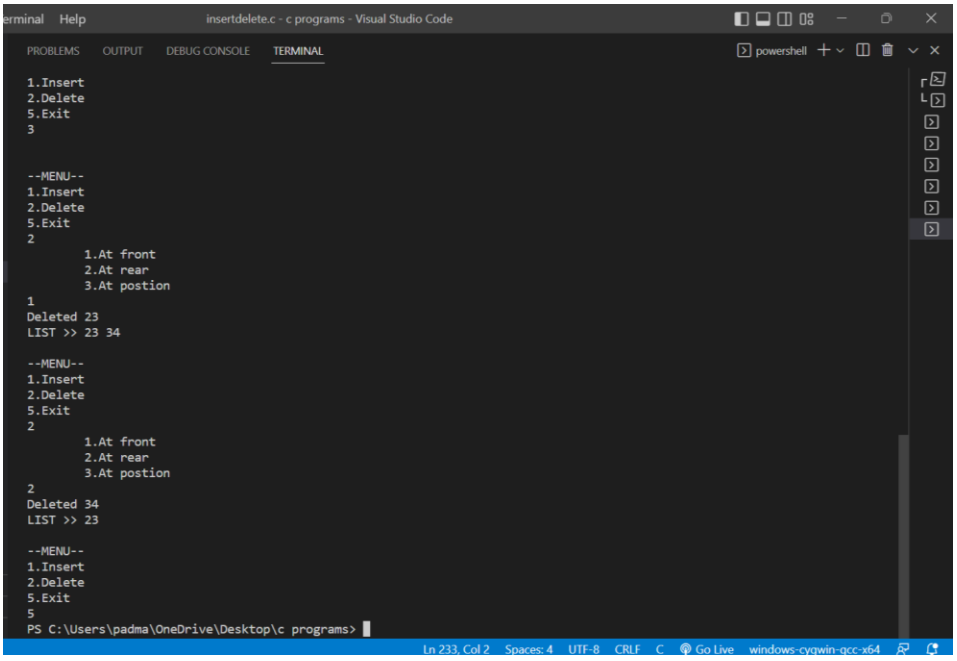
```
    return head;
}

void display(NODE head){
    NODE p;
    if(head == NULL){
        printf("List is empty\n");
        exit(0);
    }
    p = head;
    printf("LIST >> ");
    while(p != NULL){
        printf("%d ", p->data);
        p = p->next;
    }
}

void main()
{
    NODE head = NULL;
    int c, c1, ele, pos, value;
    while(1){
        printf("\n\n--MENU--\n");
        printf("1.Insert\n2.Delete\n5.Exit\n");
        scanf("%d", &c);
        switch(c){
            case 1:
```

```
            printf("\t1.At front\n\t2.At rear\n\t3.At postion\n");
            scanf("%d", &c1);
            switch(c1){
                case 1:
                    printf("Element to insert:");
                    scanf("%d", &ele);
                    head = insertFront(head, ele);
                    display(head);
                    break;
                case 2:
                    printf("Element to insert:");
                    scanf("%d", &ele);
                    head = insertRear(head, ele);
                    display(head);
                    break;
                case 3:
                    printf("Enter positon and value:");
                    scanf("%d %d", &pos, &value);
                    head = insertPos(head, value, pos);
                    display(head);
                    break;
                default:
                    printf("Wrong choice!");
            }
            break;

        case 2:
```

```
                printf("\t1.At front\n\t2.At rear\n\t3.At postion\n");
                scanf("%d", &c1);
                switch(c1){
                    case 1:
                        head = deleteFront(head);
                        display(head);
                        break;
                    case 2:
                        head = deleteRear(head);
                        display(head);
                        break;
                    case 3:
                        printf("Enter positon:");
                        scanf("%d", &pos);
                        head = deletePos(head, pos-1);
                        display(head);
                        break;
                    default:
                        printf("Wrong choice!");
                }
                break;

            case 5:
                exit(1);
        }
    }
}
```

OUTPUT:


Terminal Help
insertdelete.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
--MENU--
1.Insert
2.Delete
5.Exit
1
1.At front
2.At rear
3.At postion
2
Element to insert:23
LIST >> 23 23
--MENU--
1.Insert
2.Delete
5.Exit
1
1.At front
2.At rear
3.At postion
2
Element to insert:34
LIST >> 23 23 34
--MENU--
1.Insert
2.Delete
5.Exit
3
--MENU--
1.Insert
2.Delete
5.Exit
Ln 233, Col 2 Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64


Help
insertdelete.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
powershell
1.Insert
2.Delete
5.Exit
3
--MENU--
1.Insert
2.Delete
5.Exit
2
1.At front
2.At rear
3.At postion
1
Deleted 23
LIST >> 23 34
--MENU--
1.Insert
2.Delete
5.Exit
2
1.At front
2.At rear
3.At postion
2
Deleted 34
LIST >> 23
--MENU--
1.Insert
2.Delete
5.Exit
5
PS C:\Users\padma\OneDrive\Desktop\c programs>
Ln 233, Col 2 Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64

LAB PROGRAM 7:

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

CODE:

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


typedef struct node
{
  int data;
  struct node *next;
}*Node;


Node getnode(){
  Node x;
  x=(Node) malloc (sizeof(Node));
  if(x!=NULL)
    return x;
  else{
      printf("Memory is not allocated");
      exit(0);
    }
}


Node insert_end(Node first,int item)
```

```
{
  Node temp,x;
  x=getnode();
  x->data=item;
  x->next=NULL;
  if(first==NULL)
     return x;
  temp=first;
  while(temp->next!=NULL)
     temp=temp->next;
  temp->next=x;
  return first;
}


Node sort(Node first){
    Node temp1,temp2,head;
    head=first;
    int swap;
    //second=first->next;
    if(first==NULL)
    printf("EMPTY\n");
    else{
        for(temp1=first;temp1->next!=NULL;temp1=temp1->next){
            for(temp2=temp1->next;temp2!=NULL;temp2=temp2->next){
                if(temp1->data>temp2->data){
                swap=temp1->data;
                temp1->data=temp2->data;
```

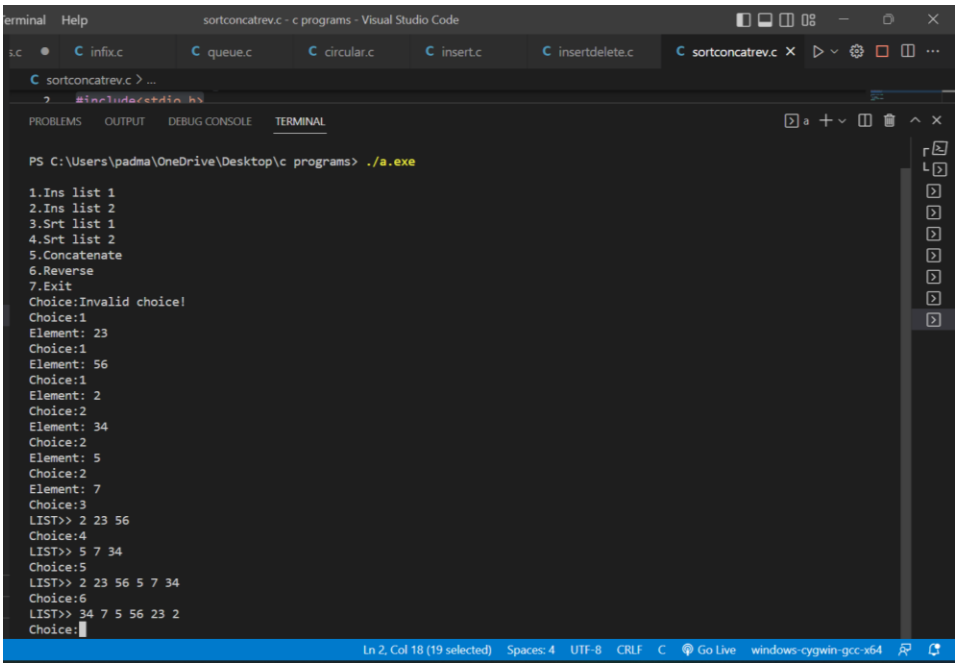
```
                temp2->data=swap;
                }
            }
        }
    }
    return head;
}

Node merging(Node first1, Node first2){
    Node temp=first1;
    while(first1->next!=NULL){
        first1=first1->next;
    }
    first1->next=first2;
    return temp;
}

Node reverse(Node first){
    Node curr,prev,temp;
    prev=NULL;
    curr=first;

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

```
    }
    first=prev;
    return  first;
}

void display(Node first){
    Node temp;
    if(first==NULL)
      printf("List is empty\n");
    else{
        temp=first;
        printf("LIST>> ");
        while(temp!=NULL){
            printf("%d ",temp->data);
            temp=temp->next;
        }
        printf("\n");
    }
}

int main(){
    Node first1=NULL;
    Node first2=NULL;
    int a,value,pos;

        printf("\n1.Ins list 1\n");
        printf("2.Ins list 2\n");
```

```
        printf("3.Srt list 1\n");
        printf("4.Srt list 2\n");
        printf("5.Concatenate\n");
        printf("6.Reverse\n");
        printf("7.Exit\n");
    while(1){
      printf("Choice:");
        scanf("%d",&a);
        switch(a){
            case 1:
                printf("Element: ");
                scanf("%d",&value);
                first1=insert_end(first1,value);
                break;

            case 2:
                printf("Element: ");
                scanf("%d",&value);
                first2=insert_end(first2,value);
                break;

            case 3:
                first1=sort(first1);
                display(first1);
                break;

            case 5:
```

```
                first1=merging(first1,first2);
                display(first1);
                break;

            case 4:
                first2=sort(first2);
                display(first2);
                break;

            case 6:
                first1=reverse(first1);
                display(first1);
                break;

            case 7:
                exit(0);
            default:
                printf("Invalid choice!\n");
        }
    }
  return 0;
}
```

OUTPUT:



LAB PROGRAM 8:

WAP to implement Stack & Queues using Linked Representation.

CODE:

STACKS USING LINKEDLIST:

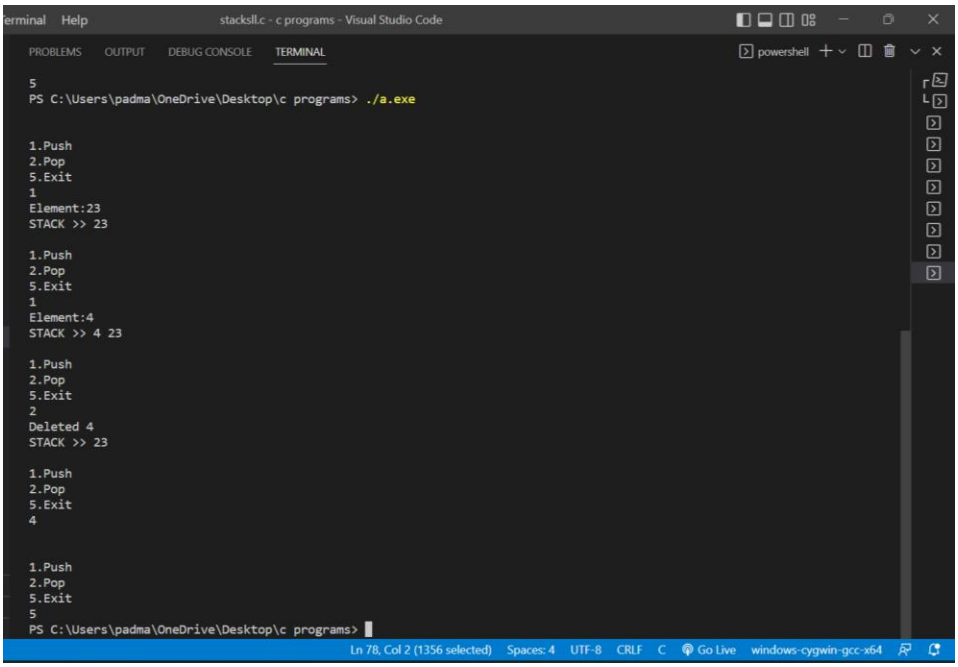
```c
#include<stdio.h>
#include<stdlib.h>


typedef struct node{
    int data;
    struct node *next;
}*NODE;



NODE getnode(){
```

```
    return (NODE)malloc(sizeof(struct node));
}



NODE insertFront(NODE head, int item){
    NODE p;
    p = getnode();
    p->data = item;
    p->next = head;
    head = p;
    return head;
}



NODE deleteFront(NODE head){
    NODE p;
    if(head == NULL)
        printf("Stack is empty");
    else{
        printf("Deleted %d\n", head->data);
        p = head->next;
        free(head);
        return p;
    }
}
```

```c
void display(NODE head){
    NODE p;
    if(head == NULL){
        printf("Stack is empty\n");
        return;
    }
    p = head;
    printf("STACK >> ");
    while(p != NULL){
        printf("%d ", p->data);
        p = p->next;
    }
}



void main()
{
    NODE head = NULL;
    int c, ele;


    while(1){
        printf("\n\n1.Push\n2.Pop\n5.Exit\n");
        scanf("%d", &c);
        switch(c){
            case 1:
                printf("Element:");
                scanf("%d", &ele);
```

```
                head = insertFront(head, ele);
                display(head);
                break;

            case 2:
                head = deleteFront(head);
                display(head);
                break;

            case 5:
                exit(1);
        }
    }
}
```

OUTPUT:



QUEUES USING LINKEDLIST:

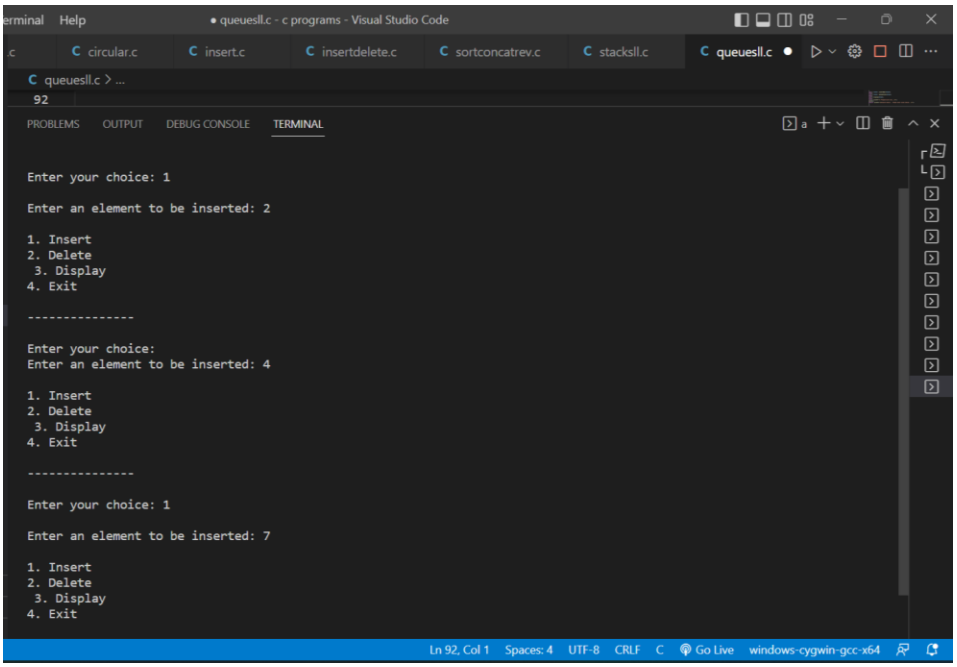
```c
#include<stdlib.h>

// Node creation

struct node

{

in}t info;

struct node *link;

};

typedef struct node *NODE;

// Insert node at the rear end

NODE insertRear(NODE first)

{

NODE temp,cur;

int item;

temp = (NODE)malloc(sizeof(struct node));

if(temp==NULL)
```

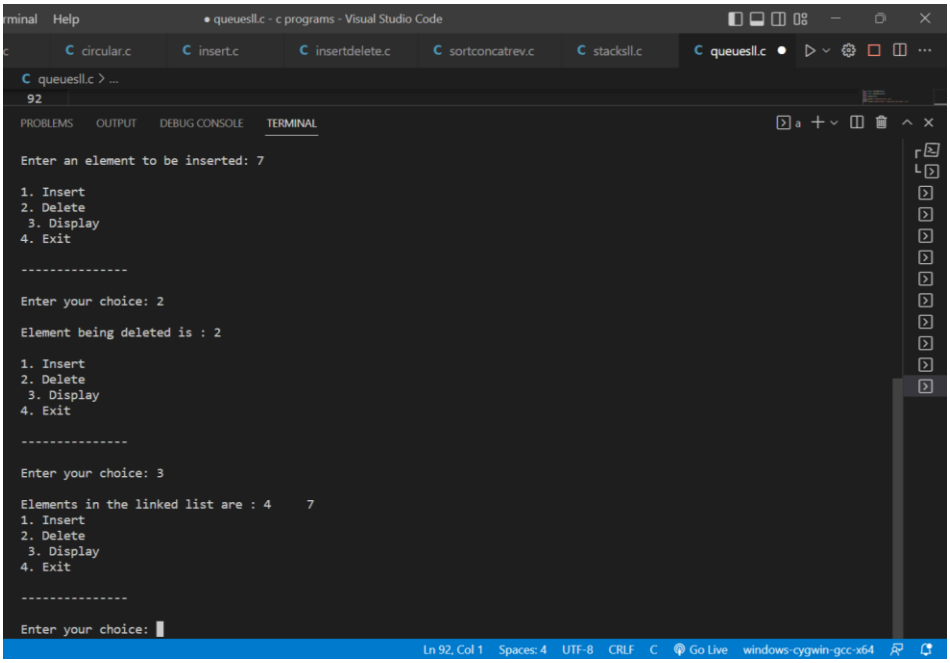
```
{
printf("\n Unable to allocate memory...\n");
return first;
}
printf("\nEnter an element to be inserted: ");
scanf("%d", &item);
temp->info = item;
temp->link = NULL;
if(first == NULL)
first=temp;
else
{
cur=first;
while(cur->link!=NULL)
cur=cur->link;
cur->link=temp;
}
return first;
}
// Delete node from the front end
NODE deleteFront(NODE first)
{
NODE temp;
if(first==NULL)
{
printf("\nThe list is empty.. no elements to delete...\n");
return first;
```

```
}
temp=first;
first = first->link;
printf("\nElement being deleted is : %d\n", temp->info);
free(temp);
return first;
}
// Display the contents of the linked list
void display(NODE first)
{
NODE cur;
if(first == NULL)
printf("\nList is empty, no elements to display\n");
else
{
cur=first;
printf("\nElements in the linked list are : ");
while(cur!=NULL)
{
printf("%d\t",cur->info);
cur=cur->link;
}
}
}
void main()
{
NODE first = NULL;
```

```
int choice;
printf("\n********* Queue implementation using SLL *********\n");
while(1)
{
printf("\n1. Insert\n2. Delete\n 3. Display\n4. Exit\n");
printf("\n---------------\n");
printf("\nEnter your choice: ");
scanf("%d",&choice);
switch(choice)
{
case 1: first = insertRear(first);
break;
case 2: first = deleteFront(first);
break;
case 3: display(first);
break;
case 4: printf("\n Program exits now...\n");
exit(0);
default: printf("\nInvalid choice... Please enter valid choice...\n");
}
}
```

OUTPUT:





LAB PROGRAM9:

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.

CODE:

```
// C program for the all operations in

// the Doubly Linked List

#include <stdio.h>

#include <stdlib.h>


// Linked List Node

struct node {

    int info;

    struct node *prev, *next;

};

struct node* start = NULL;


// Function to traverse the linked list

void traverse()

{

    // List is empty

    if (start == NULL) {

        printf("\nList is empty\n");

        return;

    }

    // Else print the Data

    struct node* temp;
```

```
    temp = start;
    while (temp != NULL) {
        printf("Data = %d\n", temp->info);
        temp = temp->next;
    }
}

// Function to insert at the front
// of the linked list
void insertAtFront()
{
    int data;
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    temp->info = data;
    temp->prev = NULL;

    // Pointer of temp will be
    // assigned to start
    temp->next = start;
    start = temp;
}

// Function to insert at the end of
// the linked list
```

```c
void insertAtEnd()

{

    int data;

    struct node *temp, *trav;

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

    temp->prev = NULL;

    temp->next = NULL;

    printf("\nEnter number to be inserted: ");

    scanf("%d", &data);

    temp->info = data;

    temp->next = NULL;

    trav = start;


    // If start is NULL

    if (start == NULL) {


        start = temp;

    }


    // Changes Links

    else {

        while (trav->next != NULL)

            trav = trav->next;

        temp->prev = trav;

        trav->next = temp;

    }

}
```

```
// Function to insert at any specified
// position in the linked list
void insertAtPosition()
{
    int data, pos, i = 1;
    struct node *temp, *newnode;
    newnode = malloc(sizeof(struct node));
    newnode->next = NULL;
    newnode->prev = NULL;

    // Enter the position and data
    printf("\nEnter position : ");
    scanf("%d", &pos);


    // If start==NULL,
    if (start == NULL) {
        start = newnode;
        newnode->prev = NULL;
        newnode->next = NULL;
    }

    // If position==1,
    else if (pos == 1) {
    // this is author method its correct but we can simply call
insertAtfront() function for this special case
```

```
    /* newnode->next = start;
        newnode->next->prev = newnode;
        newnode->prev = NULL;
        start = newnode; */
    // now this is improved by Jay Ghughriwala on geeksforgeeks
    insertAtFront();
    }

    // Change links
    else {
    printf("\nEnter number to be inserted: ");
    scanf("%d", &data);
    newnode->info = data;
    temp = start;
        while (i < pos - 1) {
            temp = temp->next;
            i++;
        }
        newnode->next = temp->next;
        newnode->prev = temp;
        temp->next = newnode;
        temp->next->prev = newnode;
    }
}

// Function to delete from the front
// of the linked list
```

```
void deleteFirst()

{

    struct node* temp;

    if (start == NULL)

        printf("\nList is empty\n");

    else {

        temp = start;

        start = start->next;

        if (start != NULL)

            start->prev = NULL;

        free(temp);

    }

}


// Function to delete from the end

// of the linked list

void deleteEnd()

{

    struct node* temp;

    if (start == NULL)

        printf("\nList is empty\n");

    temp = start;

    while (temp->next != NULL)

        temp = temp->next;

    if (start->next == NULL)

        start = NULL;

    else {
```

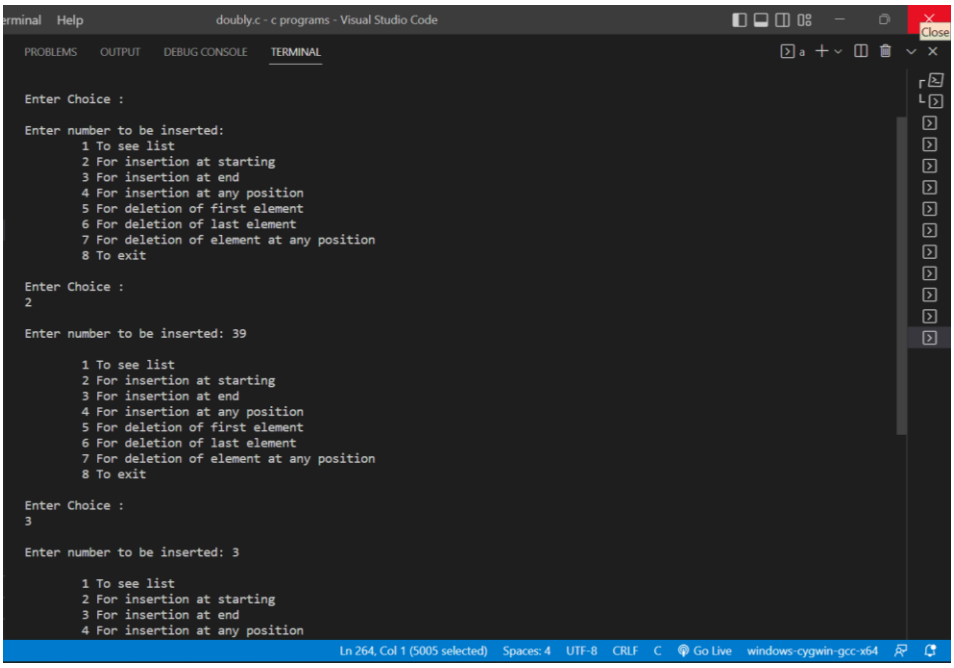
```
        temp->prev->next = NULL;
        free(temp);
    }
}

// Function to delete from any specified
// position from the linked list
void deletePosition()
{
    int pos, i = 1;
    struct node *temp, *position;
    temp = start;

    // If DLL is empty
    if (start == NULL)
        printf("\nList is empty\n");

    // Otherwise
    else {
        // Position to be deleted
        printf("\nEnter position : ");
        scanf("%d", &pos);

        // If the position is the first node
        if (pos == 1) {
            deleteFirst(); // im,proved by Jay Ghughriwala on
GeeksforGeeks
```

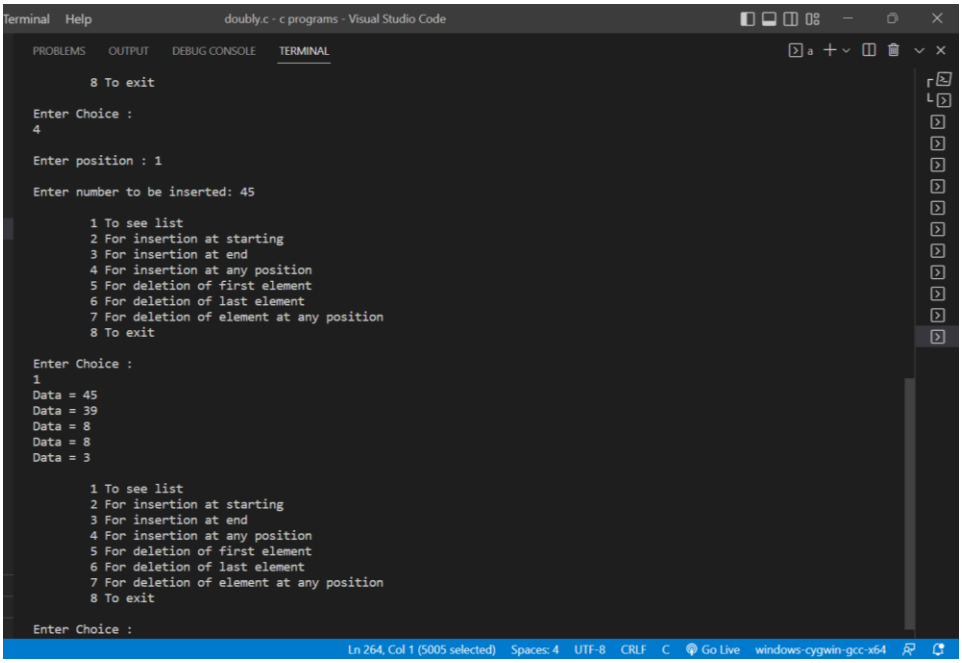
```
            if (start != NULL) {
                start->prev = NULL;
            }
            free(position);
            return;
        }

        // Traverse till position
        while (i < pos - 1) {
            temp = temp->next;
            i++;
        }
        // Change Links
        position = temp->next;
        if (position->next != NULL)
            position->next->prev = temp;
        temp->next = position->next;

        // Free memory
        free(position);
    }
}

// Driver Code
int main()
{
    int choice;
```

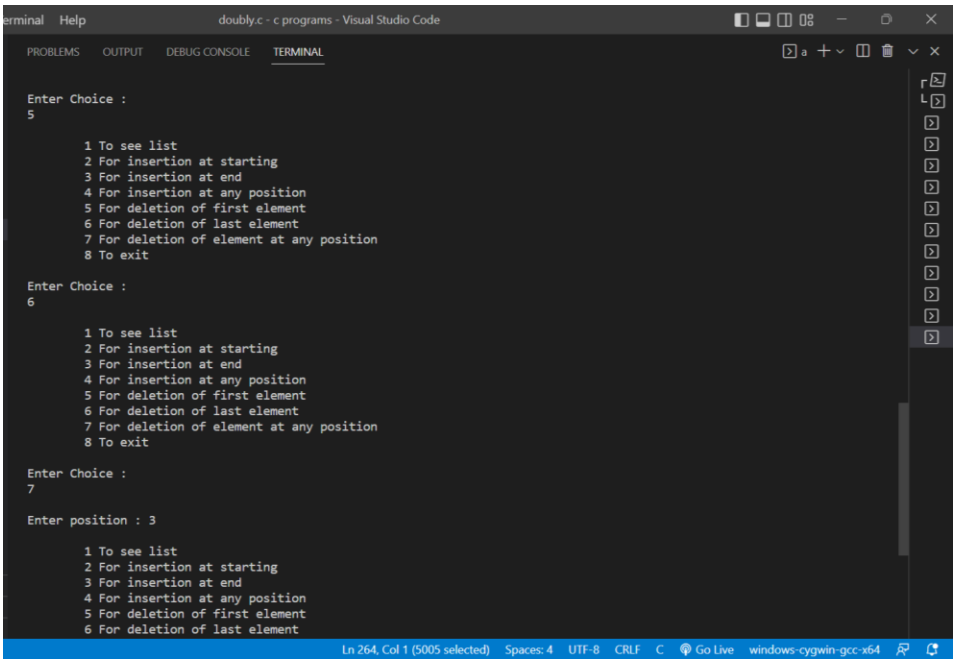
```
    while (1) {


        printf("\n\t1 To see list\n");
        printf("\t2 For insertion at"
            " starting\n");
        printf("\t3 For insertion at"
            " end\n");
        printf("\t4 For insertion at "
            "any position\n");
        printf("\t5 For deletion of "
            "first element\n");
        printf("\t6 For deletion of "
            "last element\n");
        printf("\t7 For deletion of "
            "element at any position\n");
        printf("\t8 To exit\n");
        printf("\nEnter Choice :\n");
        scanf("%d", &choice);


        switch (choice) {
        case 1:
            traverse();
            break;
        case 2:
            insertAtFront();
            break;
        case 3:
```

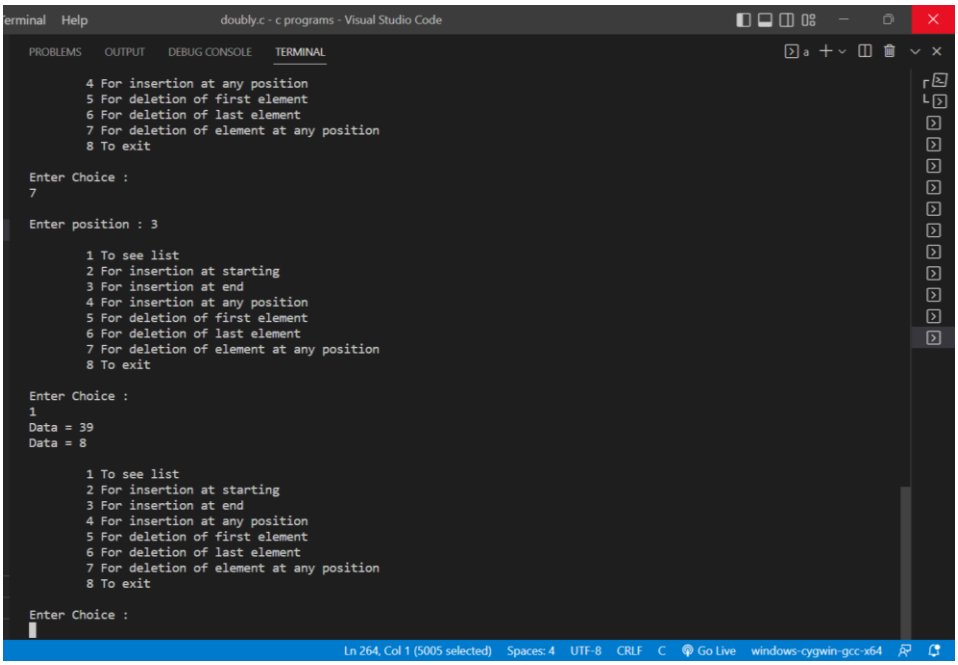
```
            insertAtEnd();
            break;
        case 4:
            insertAtPosition();
            break;
        case 5:
            deleteFirst();
            break;
        case 6:
            deleteEnd();
            break;
        case 7:
            deletePosition();
            break;

        case 8:
            exit(1);
            break;
        default:
            printf("Incorrect Choice. Try Again \n");
            continue;
        }
    }
    return 0;
}
```

OUTPUT:


rminal Help
doubly.c - c programs - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
Enter Choice :
Enter number to be inserted:
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
2
Enter number to be inserted: 39
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
3
Enter number to be inserted: 3
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
Ln 264, Col 1 (5005 selected)
Spaces: 4
UTF-8
CRLF
C
Go Live
windows-cygwin-gcc-x64


Terminal Help
doubly.c - c programs - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
8 To exit
Enter Choice :
4
Enter position : 1
Enter number to be inserted: 45
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
1
Data = 45
Data = 39
Data = 8
Data = 8
Data = 3
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
Ln 264, Col 1 (5005 selected)
Spaces: 4
UTF-8
CRLF
C
Go Live
windows-cygwin-gcc-x64


erminal Help
doubly.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Enter Choice :
5
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
6
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
7
Enter position : 3
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
Ln 264, Col 1 (5005 selected) Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64


erminal Help
doubly.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
7
Enter position : 3
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
1
Data = 39
Data = 8
1 To see list
2 For insertion at starting
3 For insertion at end
4 For insertion at any position
5 For deletion of first element
6 For deletion of last element
7 For deletion of element at any position
8 To exit
Enter Choice :
Ln 264, Col 1 (5005 selected) Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64

LAB PROGRAM10:

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.

CODE:

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


typedef struct Node{
    struct Node *left;
    int data;
    struct Node *right;
}*node;



node getnode(int item){
    node temp = (node)malloc(sizeof(struct Node));

    temp->left = NULL;
    temp->data = item;
    temp->right = NULL;

    return temp;
}
```

```
node insert(node root, int ele){
    if(root == NULL)
        return getnode(ele);
    else if(ele < root->data)
        root->left = insert(root->left, ele);
    else if(ele > root->data)
        root->right = insert(root->right, ele);

    return root;
}



void inorder(node root){
    if(root == NULL)
        return;

    inorder(root->left);
    printf("%d ", root->data);
    inorder(root->right);
}



void preorder(node root){
    if(root == NULL)
        return;

    printf("%d ", root->data);
```

```
    preorder(root->left);
    preorder(root->right);
}



void postorder(node root){
    if(root == NULL)
        return;

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



int search(node root, int ele){
    if(root == NULL)
        return 0;

    node temp = root;
    while(temp != NULL){
        if(ele == temp->data)
            return 1;
        else if(ele < temp->data)
            temp = temp->left;
        else // if ele > temp->data
            temp = temp->right;
```

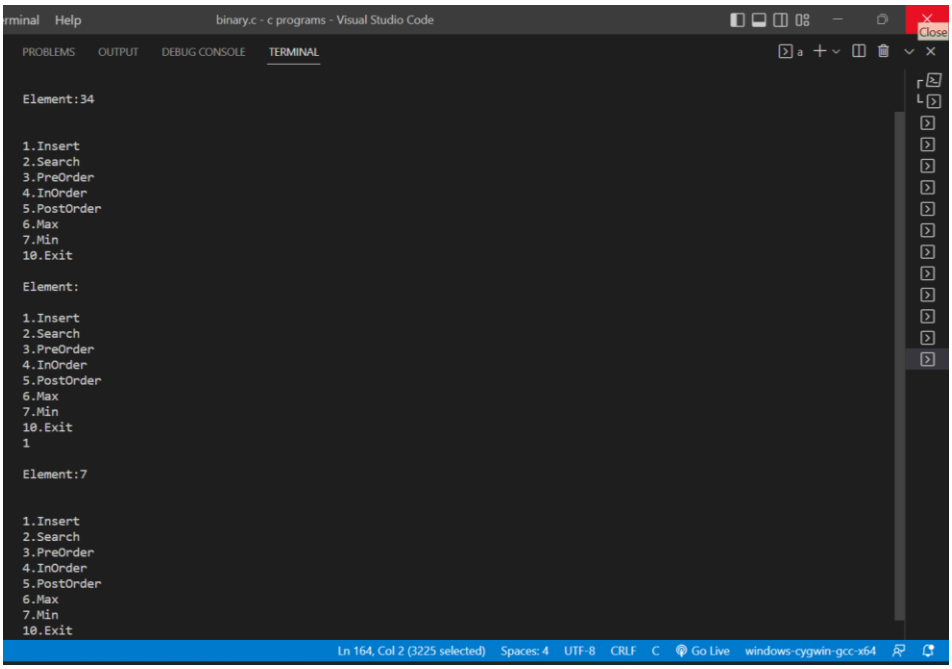
```
    }
    // if not found within while loop, ret 0;
    return 0;
}



int max(node root){
    if(root == NULL)
        return -1;

    int maxval;
    node temp = root;
    while(root != NULL){
        maxval = root->data;
        root = root->right;
    }
    return maxval;
}



int min(node root){
    if(root == NULL)
        return -1;

    int minval;
    node temp = root;
    while(root != NULL){
```

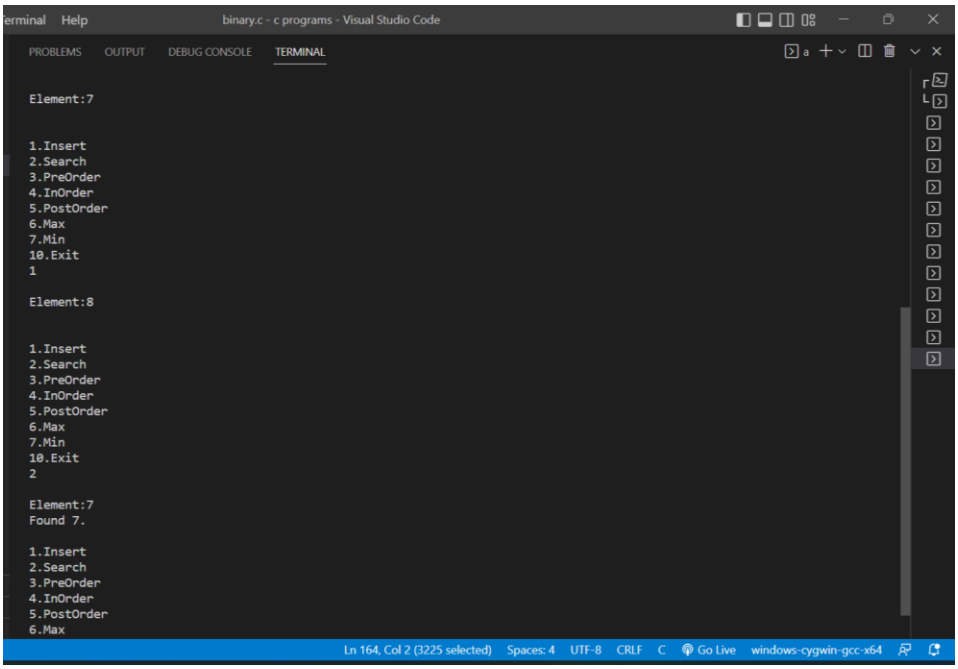
```
        minval = root->data;
        root = root->left;
    }
    return minval;
}



void main(){
    node root = NULL;
    int e, ch = 1;

    while(ch!=10){

printf("\n\n1.Insert\n2.Search\n3.PreOrder\n4.InOrder\n5.PostOrder\n");
        printf("6.Max\n7.Min\n10.Exit\n");
        scanf("%d", &ch);
        printf("\n");

        switch(ch){
            case 1:
                printf("Element:");
                scanf("%d", &e);
                root = insert(root, e);
                break;

            case 2:
                printf("Element:");
```

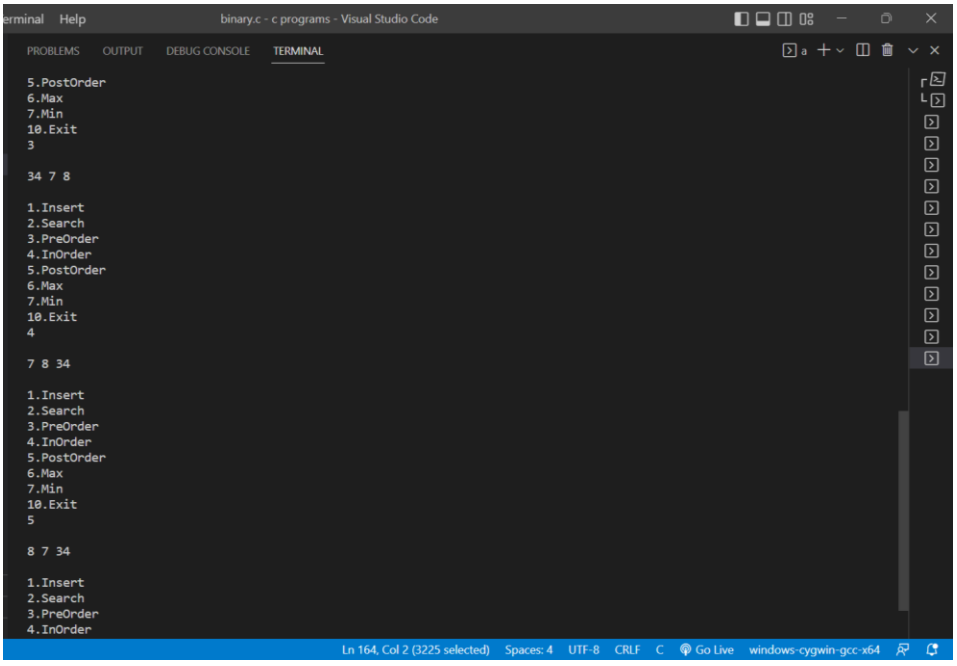
```
            scanf("%d", &e);
            if(search(root, e))
                printf("Found %d.", e);
            else
                printf("Couldn't find %d.", e);
            break;

        case 3:
            preorder(root);
            break;

        case 4:
            inorder(root);
            break;

        case 5:
            postorder(root);
            break;

        case 6:
            printf("Max: %d", max(root));
            break;

        case 7:
            printf("Min: %d", min(root));
            break;
```

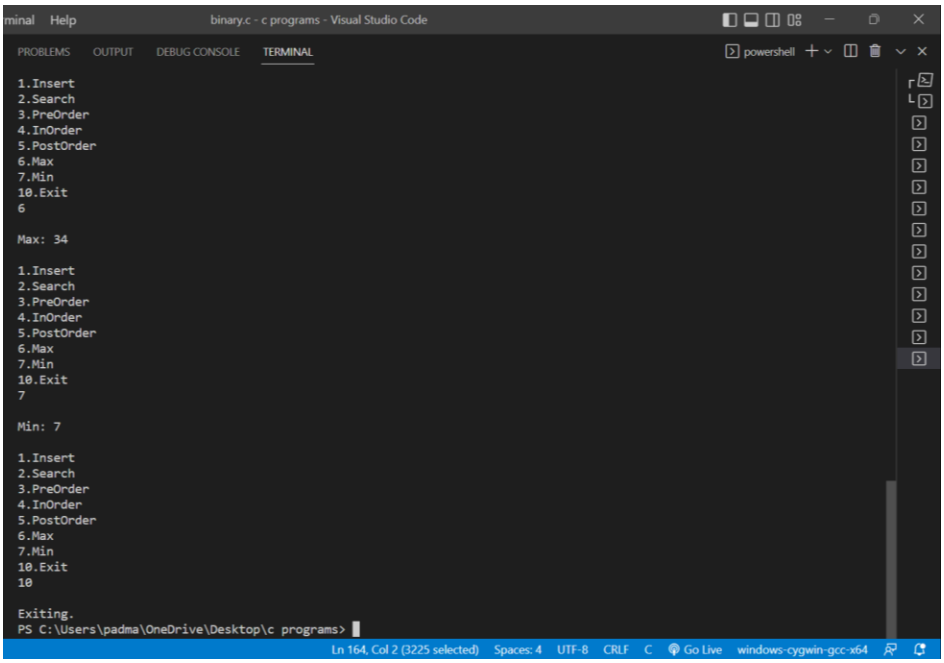
```
            case 10:
                printf("Exiting.");
                exit(1);

            default:
                printf("Wrong input!");
        }
    }
}
```

OUTPUT:




erminal Help
binary.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Element:7
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
1
Element:8
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
2
Element:7
Found 7.
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
Ln 164, Col 2 (3225 selected) Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64


erminal Help
binary.c - c programs - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
5.PostOrder
6.Max
7.Min
10.Exit
3
34 7 8
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
4
7 8 34
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
5
8 7 34
1.Insert
2.Search
3.PreOrder
4.InOrder
Ln 164, Col 2 (3225 selected) Spaces: 4 UTF-8 CRLF C Go Live windows-cygwin-gcc-x64


minal Help
binary.c - c programs - Visual Studio Code
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
powershell
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
6
Max: 34
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
7
Min: 7
1.Insert
2.Search
3.PreOrder
4.InOrder
5.PostOrder
6.Max
7.Min
10.Exit
10
Exiting.
PS C:\Users\padma\OneDrive\Desktop\c programs>
Ln 164, Col 2 (3225 selected)
Spaces: 4
UTF-8
CRLF
C
Go Live
windows-cygwin-gcc-x64