|  |
| --- |
| NAME : SARASWATHI B |
| USN : 1BM19CS032 |
| DEPARTMENT : CSE |
| SUBJECT : DATA STRUCTURES LAB |
| ACADEMIC YEAR : 2020-21 |

**LAB-1**

Write a program to simulate the working of stack using an array with the following : a) Push b) Pop c) Display. The program should print appropriate messages for stack overflow, stack underflow

**PROGRAM**

#include<stdio.h>

int stack[100],choice,n,top,x,i;

void push(void);

void pop(void);

void display(void);

int main()

{

    top=-1;

    printf("\n Enter the size of STACK[MAX=100]:");

    scanf("%d",&n);

    printf("\n\t STACK OPERATIONS USING ARRAY");

    printf("\n\t--------------------------------");

    printf("\n\t 1.PUSH\n\t 2.POP\n\t 3.DISPLAY\n\t 4.EXIT");

    do

    {

        printf("\n Enter the Choice:");

        scanf("%d",&choice);

        switch(choice)

        {

            case 1:

            {

                push();

                break;

            }

            case 2:

            {

                pop();

                break;

            }

            case 3:

            {

                display();

                break;

            }

            case 4:

            {

                printf("\n\t EXIT POINT ");

                break;

            }

            default:

            {

                printf ("\n\t Please Enter a Valid Choice(1/2/3/4)");

            }

        }

    }

    while(choice!=4);

    return 0;

}

void push()

{

    if(top>=n-1)

    {

        printf("\n\tSTACK is over flow");

    }

    else

    {

        printf(" Enter a value to be pushed:");

        scanf("%d",&x);

        top++;

        stack[top]=x;

    }

}

void pop()

{

    if(top<=-1)

    {

        printf("\n\t Stack is under flow");

    }

    else

    {

        printf("\n\t The popped elements is %d",stack[top]);

        top--;

    }

}

void display()

{

    if(top>=0)

    {

        printf("\n The elements in STACK \n");

        for(i=top; i>=0; i--)

            printf("\n%d",stack[i]);

        printf("\n Press Next Choice");

    }

    else

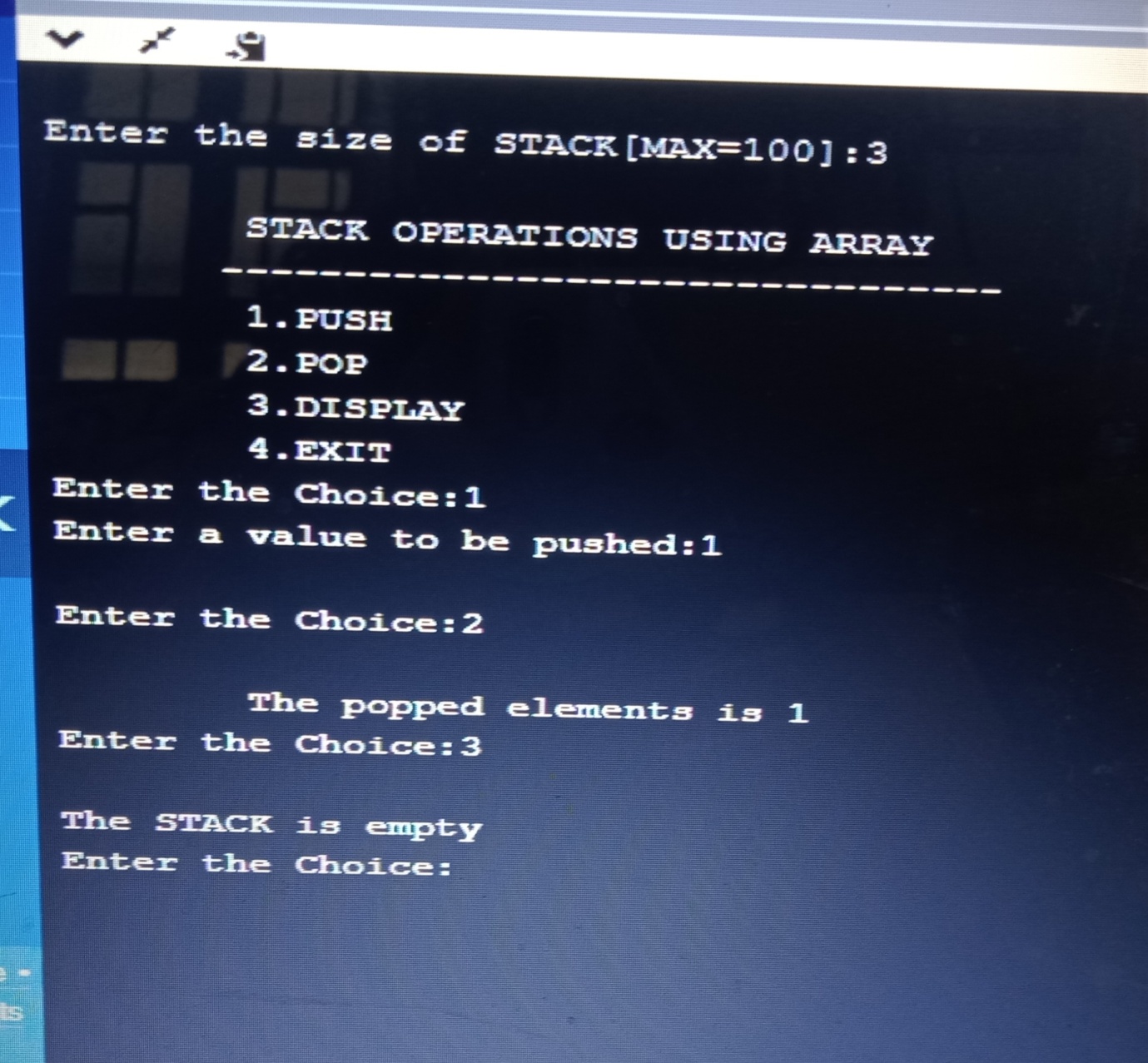
    {

        printf("\n The STACK is empty");

    }

}

**OUTPUT**



**LAB-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>

# define MAX 100

char stack[MAX];

int top=-1;

void push(char ch)

{

    if (top==MAX-1)

        printf("Stack is full\n");

    else

    {

        top++;

        stack[top]=ch;

    }

}

char pop()

{

    char item;

    if (top==-1)

        printf("\n stack is empty");

    else

    {

        item=stack[top];

        top--;

        return item;

    }

}

int stackempty()

{

    if(top==-1) return 1;

    else return 0;

}

char stacktop()

{

    if( top==-1)

        printf("\n stack is empty!");

    else

        return stack[top];

}

int priority(char ch)

{

    switch(ch)

    {

        case '+':

        case '-':return (1);

        case '\*':

        case '/':return (2);

        default : return (0);

    }

}

int main(int argc, char \*\*argv)

{

char infix[100];

    int i, item;

    printf("Enter the infix expression :");

    scanf("%s",infix);

    printf("Expression : %s",infix);

    printf("\n Postfix: ");

    i=0;

     while (infix[i]!='\0')

    {

        switch (infix[i])

        {

            case '(': push(infix[i]);

                      break;

            case ')':while(( item=pop())!='(')

                        printf("%c",item);

                      break;

            case '+':

            case '-':

            case '\*':

            case '/':

                      while(!stackempty() && priority(infix[i])<=priority(stacktop()))

                      {

                            item=pop();

                            printf("%c", item);

                      }

                      push(infix[i]);

                      break;

            default : printf("%c", infix[i]);

                      break;

        }

         i++;

    }

    while(!stackempty())

    {

        char item;

        item=pop();

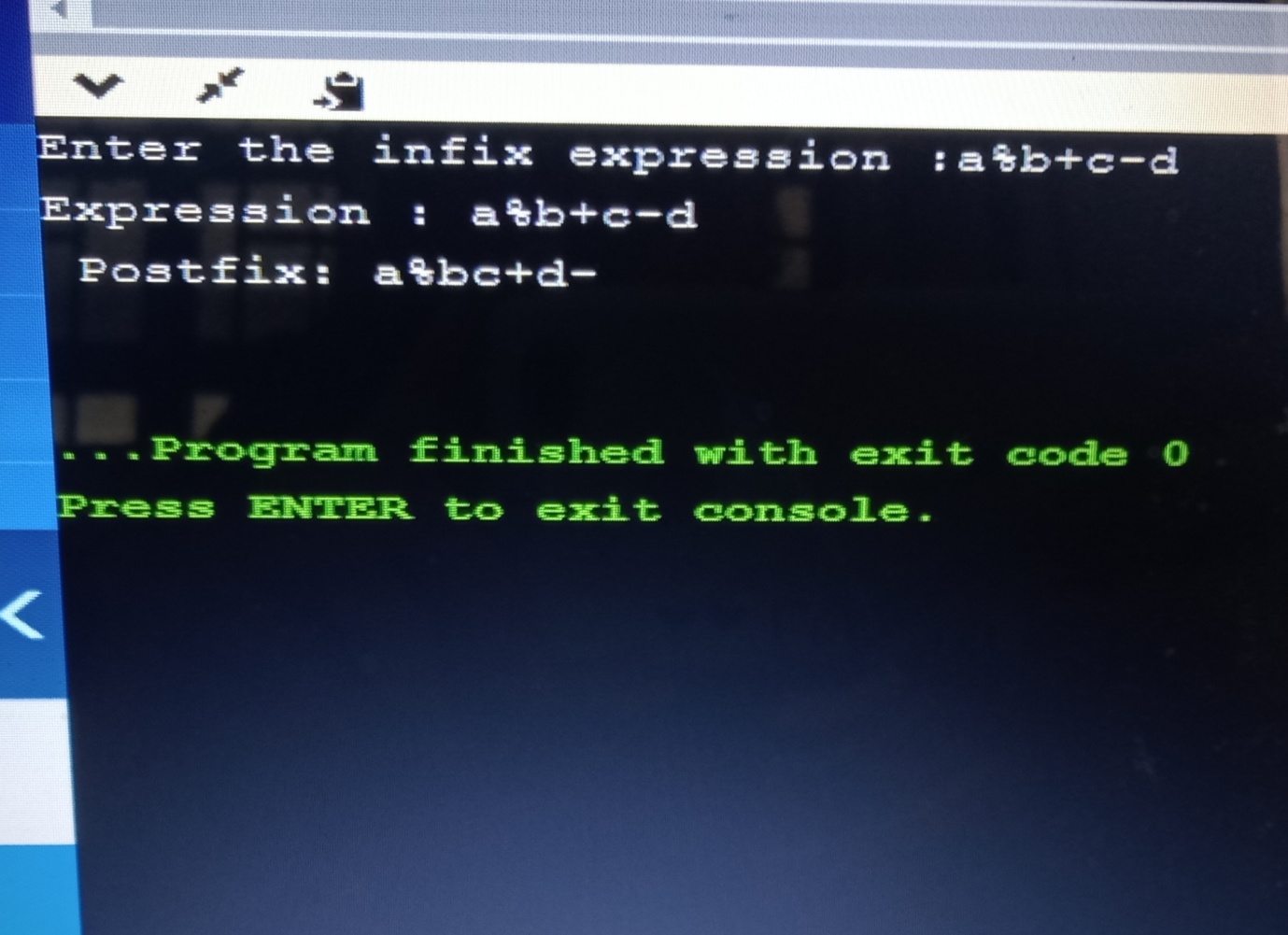
        printf("%c", item);

    }

    printf("\n");

    return 0;

}



**LAB-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

#include <stdio.h>

#include <stdlib.h>

#define MAX 5

int front=0;

int rear=-1;

int queue[MAX];

void Enque(int);

int Deque();

void display();

int main(int argc, char \*\*argv)

{

int option;

    int item;

    do{

        printf("\n 1. Insert to Queue (EnQueue)");

        printf("\n 2. delete from the Queue (DeQueue)");

        printf("\n 3. Display the content ");

        printf("\n 4. Exit\n");

        printf("Enter the option :");

        scanf("%d",&option);

        switch(option)

        {

            case 1: printf("Enter the element\n");

                     scanf("%d",&item);

                     Enque(item);

                     break;

            case 2: item=Deque();

                    if(item==-1)

                        printf("Queue is empty\n");

                    else

                    printf("Removed element from the queue %d",item);

                    break;

            case 3: display();

                    break;

            case 4: exit(0);

        }

    } while (option!=4);

return 0;

}

void Enque(int ele)

{

    if (rear==MAX-1)

       printf("Queue is full\n");

    else

    {

      rear++;

      queue[rear]=ele;

    }

}

int Deque()

{

    int item;

    if(front == -1)

        return -1;

    else

    {

        item=queue[front];

        front++;

        if(front>rear)

        {

            front=-1;

            rear=-1;

        }

        return item;

    }

}

void display()

{

    int i;

    if(front==-1)

        printf("Queue is empty\n");

    else

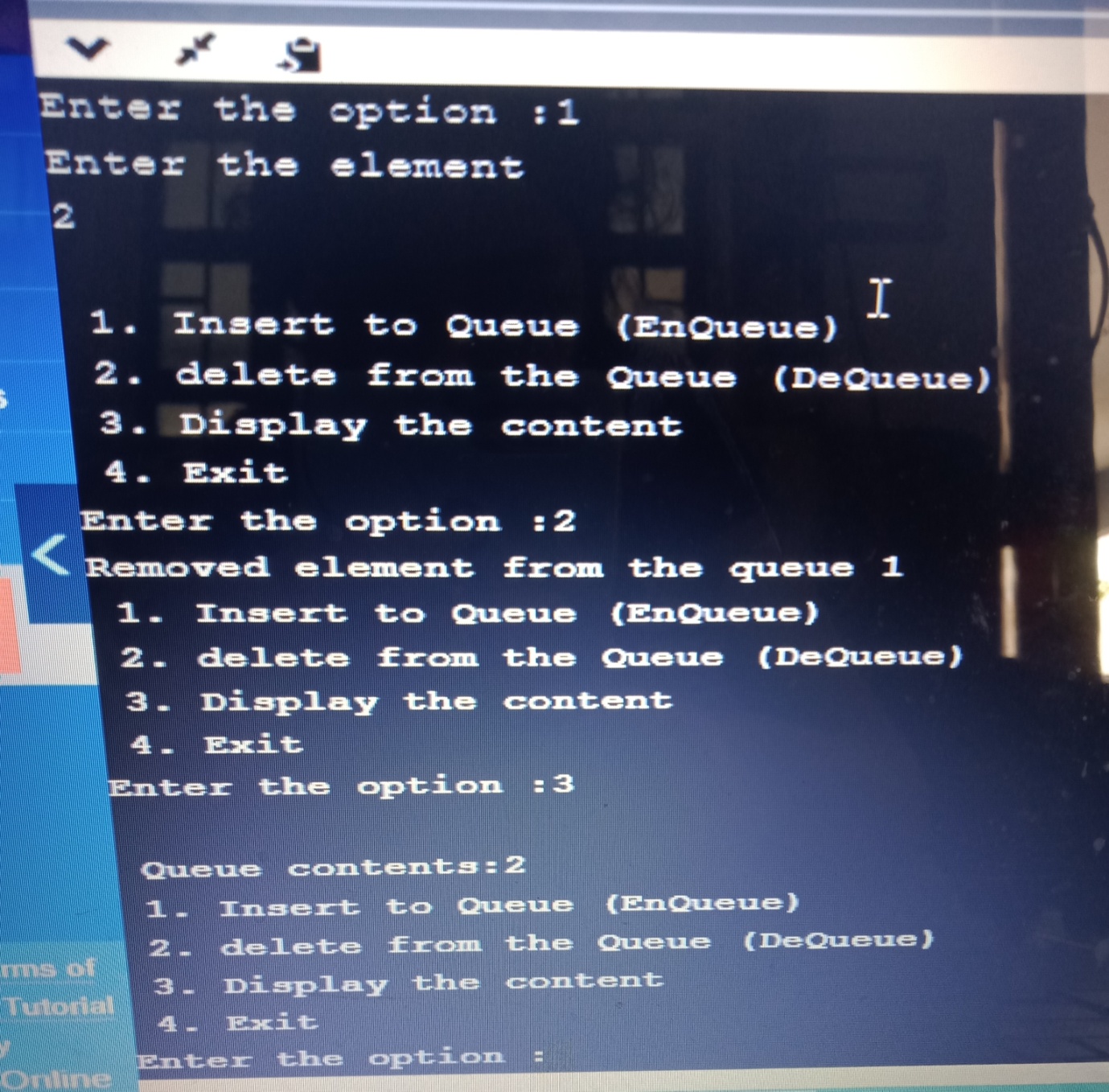
    {

        printf("\n Queue contents:");

        for(i=front;i<=rear;i++)

            printf("%d", queue[i]);

    }



**LAB-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

#include <stdio.h>

#include <stdlib.h>

#define MAX 3

int front=-1;

int rear=-1;

int queue[MAX];

void Enque(int);

int Deque();

void display();

int main(int argc, char \*\*argv)

{

int option;

    int item;

    do{

        printf("Circular Queue\n");

        printf("\n 1. Insert to Queue (EnQueue)");

        printf("\n 2. delete from the Queue (DeQueue)");

        printf("\n 3. Display the content ");

        printf("\n 4. Exit\n");

        printf("Enter the option :");

        scanf("%d",&option);

        switch(option)

        {

            case 1: printf("Enter the element\n");

                     scanf("%d",&item);

                     Enque(item);

                     break;

            case 2: item=Deque();

                    if(item==-999)

                        printf("Queue is empty");

                    else

                    printf("Removed element from the queue %d",item);

                    break;

            case 3: display();

                    break;

            case 4: exit(0);

        }

    } while (option!=4);

return 0;

}

void Enque(int ele)

{

    if(((front == 0 && rear == MAX - 1))|| (front == rear + 1) )

    {

       printf("Queue is full\n");return;

    }

    else

    {

      rear=(rear+1)%MAX;

      queue[rear]=ele;

      if(front ==-1)

          front=0;

    }

}

int Deque()

{

    int item;

    if((front == -1)&&(rear == -1))

    {

        return(-999);

    }

    else

    {

        item=queue[front];

        if(front==rear)

        {

            front=-1;

            rear=-1;

        }

        else

        {

            front=(front+1)%MAX;

        }

        return item;

    }

}

void display()

{

    int i;

    if(((front==-1)&& (rear==-1))|| (front==rear))

    {

        printf("Queue is empty\n");return;

    }

    else

    {

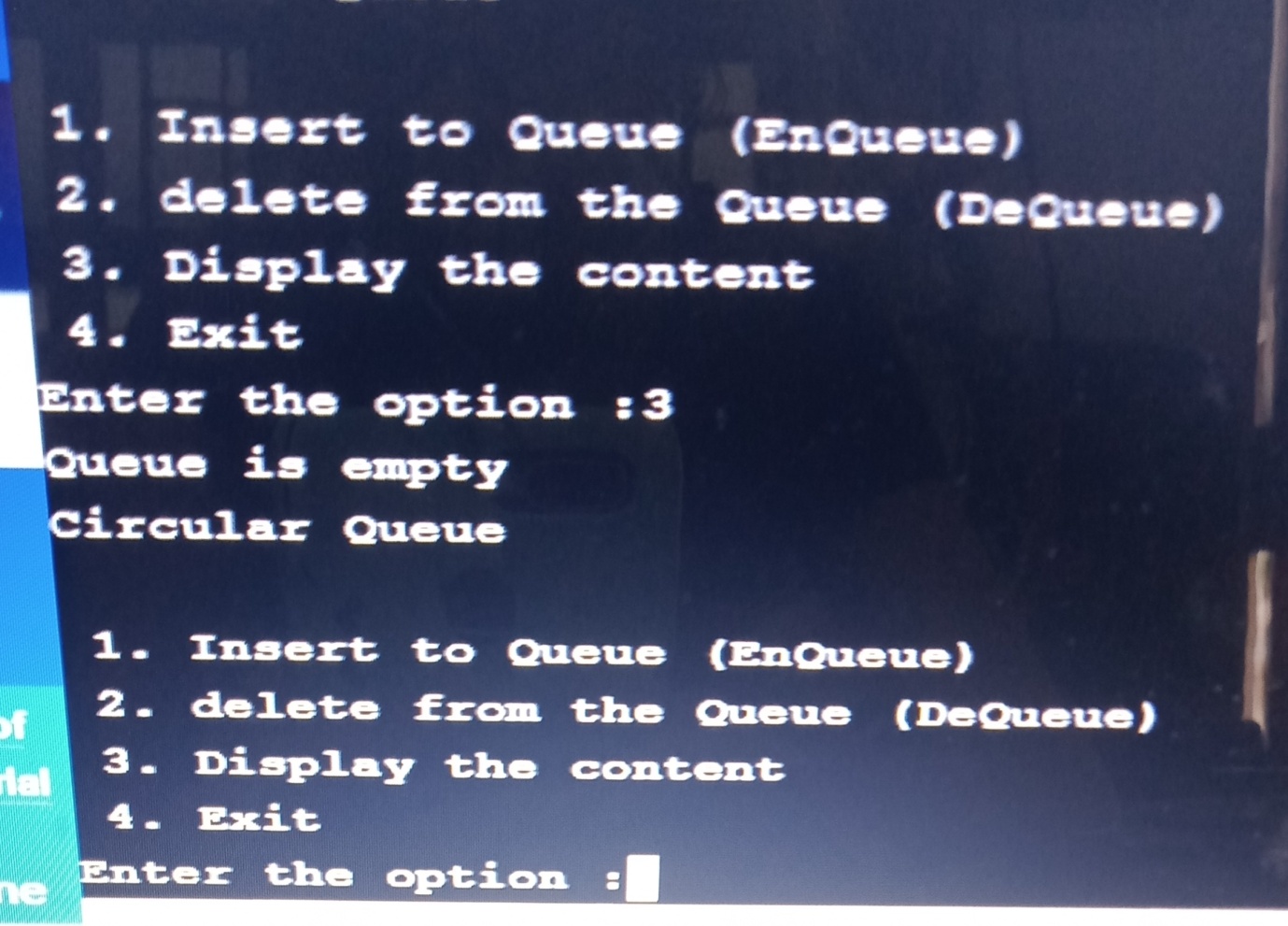
        printf("\n Queue contents:\n");

        for(i=front;i<=rear;i++)

            printf("%d", queue[i]);

    }

}



**LAB-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.

#include<stdio.h>

#include <stdlib.h>

void create();

void display();

void insert\_at\_node(int);

void insert\_before();

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

int main()

{

int choice,ele;

do

{

printf("\n1. Create \n2. Display\n3. Insert before \n4. Insert at particular position\n5.

Exit \n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1: create(); break;

case 2: display();break;

case 3: insert\_before();

break;

case 4 :printf("enter the position where new element has to be inserted\n");

scanf("%d",&ele);

insert\_at\_node(ele);

break;

default: exit(0);

}

}while(choice==1 || choice==2 || choice==3 || choice==4);

return 0;

}

void create()

{

struct node \*newnode,\*temp;

int item;

newnode =(struct node \*) malloc (sizeof(struct node));

printf("Enter the data : ");

scanf("%d",&item);

newnode->data=item;

if (head==NULL)

{

newnode->next=NULL;

head=newnode;

printf("Node created\n");

}

else

{

temp=head;//transversing

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=NULL;

printf("Node created\n");

}

}

void insert\_before()

{

struct node \*newnode;

int ele;

printf("Enter the element : ");

scanf("%d",&ele);

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data =ele;

newnode->next=head;

head=newnode;

}

void insert\_at\_node(int a)

{

struct node \*newnode,\*temp;

int i=1,ele;

printf("Enter the element : ");

scanf("%d",&ele);

temp=head;

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data =ele;

while (i<(a-1))

{

temp=temp->next;

i++;

}

newnode->next=temp->next;

temp->next=newnode;

}

void display()

{

struct node \*ptr=NULL;

ptr=head;

if(ptr==NULL)

{

printf("list empty!!!\n");

}

else

{

while(ptr!=NULL)

{

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

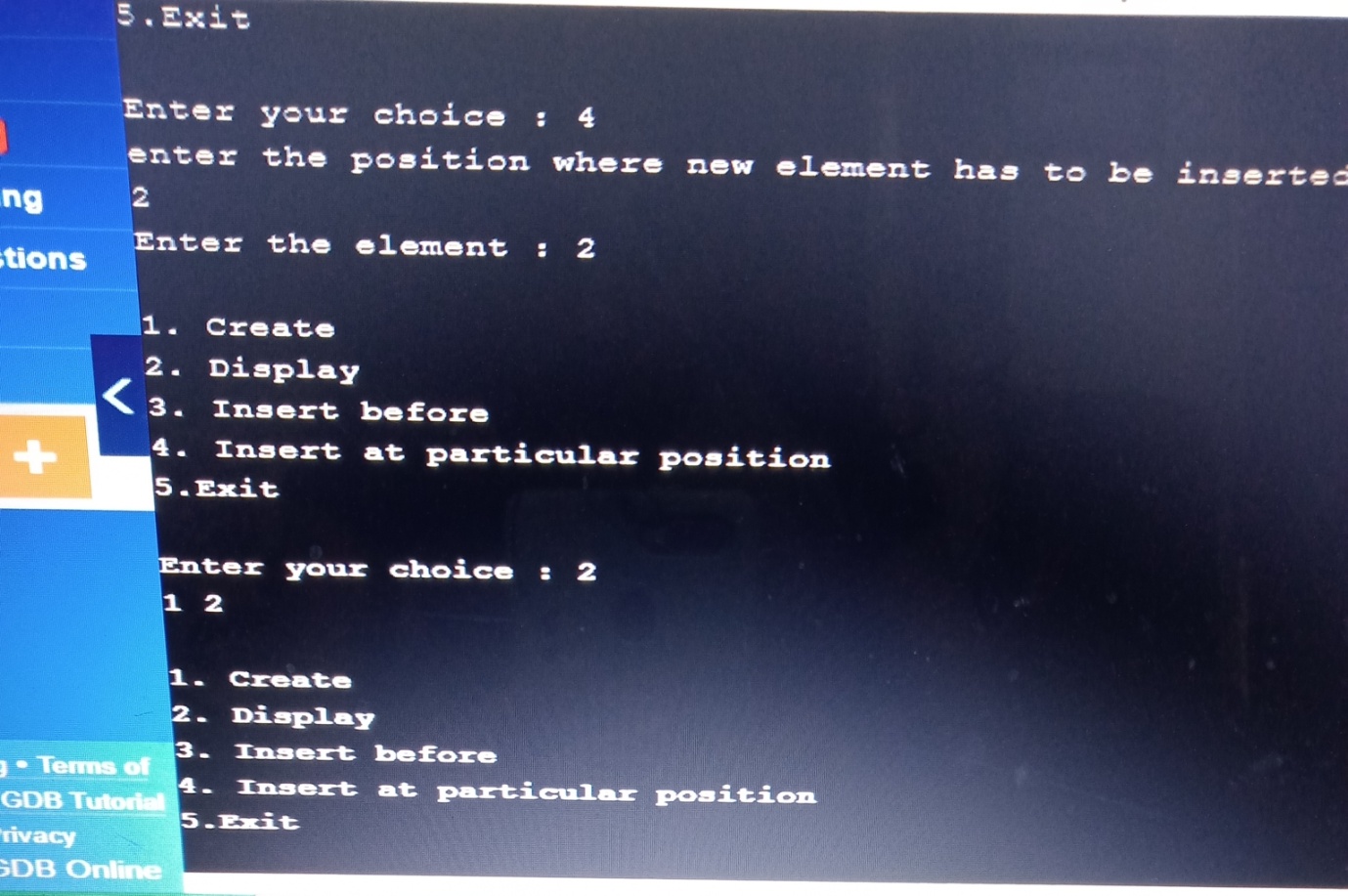
ptr=ptr->next;

}

}

printf("\n");

}



**LAB-6**

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.

#include<stdio.h>

#include <stdlib.h>

void create();

void display();

void delete\_front();

void delete\_last();

void delete\_at\_node(int);

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

int main()

{

int choice,ele;

do

{

printf("\n1. Create \n2. Display\n3. Delete at first\n4. Delete from end\n5.

Delete particular element\n6. Exit \n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1: create(); break;

case 2: display();break;

case 3: delete\_front();

break;

case 4:delete\_last();

break;

case 5:printf("Enter the element to be deleted\n");

scanf("%d",&ele);

delete\_at\_node(ele);

break;

default: exit(0);

}

}while(choice==1 || choice==2 || choice==3 || choice==4 ||choice==5);

return 0;

}

void create()

{

struct node \*newnode,\*temp;

int item;

newnode =(struct node \*) malloc (sizeof(struct node));

printf("Enter the data : ");

scanf("%d",&item);

newnode->data=item;

if (head==NULL)

{

newnode->next=NULL;

head=newnode;

printf("Node created\n");

}

else

{

temp=head;//transversing

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=NULL;

printf("Node created\n");

}

}

void delete\_front()

{

if (head == NULL)

{

printf("Empty List. Can't delete\n");return;

}

else

head=head->next;

}

void delete\_last()

{

struct node \*temp;

if (head == NULL)

{

printf("Empty List. Can't delete\n");return;

}

else

{

temp=head;

while(temp->next->next!=NULL)

{

//printf("%d",temp->data);

temp=temp->next;

}

temp->next=NULL;

}

}

void delete\_at\_node(int ele)

{

struct node \*temp,\*del=NULL;

if (head == NULL)

{

printf("Empty List. Can't delete\n");

return;

}

temp=head;

if(head->data==ele)

{

head=head->next;

return;

}

while (temp->next!=NULL)

{

if(temp->next->data==ele)

{

del=temp->next;

if(del->next==NULL)

temp->next=NULL;//deleting at end

else

temp->next=del->next;

}

else

temp=temp->next;

}

if(del==NULL)

{

printf("Element not found in the list\n");

return;

}

}

void display()

{

struct node \*ptr=NULL;

ptr=head;

if(ptr==NULL)

{

printf("list empty!!!\n");

}

else

{

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

printf("\n");

}



**LAB-7**

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

#include<stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

void create(struct node \*\*);

void display(struct node \*);

void concat(struct node \*, struct node \*);

int main(int argc, char \*\*argv)

{

struct node \*head1=NULL, \*head2=NULL;

printf("Create two list\n");

printf("Creating List one\n");

create(&head1);

printf("Creating List two\n");

create(&head2);

concat(head1,head2);

display(head1);

}

void create(struct node \*\*hptr)

{

struct node \*newnode,\*temp;

int item;

int choice=1;

do

{

newnode =(struct node \*) malloc (sizeof(struct node));

printf("Enter the data : ");

scanf("%d",&item);

newnode->data=item;

newnode->next=NULL;

printf("Do u want add element in the list:(if yes enter 1)\n");

scanf("%d", &choice);

if (\*hptr==NULL)

{

\*hptr=newnode;

}

else

{

temp=\*hptr;

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=NULL;

}

}while (choice==1);

}

void concat (struct node \*temp1, struct node \*temp2)

{

while(temp1->next!=NULL)

temp1=temp1->next;

temp1->next=temp2;

}

void display(struct node \*ptr)

{

if(ptr==NULL)

{

printf("Nothing to print\n");

}

else

{

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

printf("\n");

}

#include<stdio.h>

#include <stdlib.h>

void create();

void display();

void sort();

void reverse();

struct node

{

int data;

struct node \*next;

};

struct node \*head=NULL;

int main()

{

int choice,ele;

do

{

printf("\n1. Create \n2. Display\n3. Sort\n4. Reverse\n5. Exit \n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1: create(); break;

case 2: display();break;

case 3: sort();

break;

case 4: reverse();

break;

default: exit(0);

}

}while(choice==1 || choice==2 || choice==3 || choice==4);

return 0;

}

void create()

{

struct node \*newnode,\*temp;

int item;

newnode =(struct node \*) malloc (sizeof(struct node));

printf("Enter the data : ");

scanf("%d",&item);

newnode->data=item;

if (head==NULL)

{

newnode->next=NULL;

head=newnode;

printf("Node created\n");

}

else

{

temp=head;//transversing

while(temp->next!=NULL)

{

temp=temp->next;

}

temp->next=newnode;

newnode->next=NULL;

printf("Node created\n");

}

}

void display()

{

struct node \*ptr=NULL;

ptr=head;

if(ptr==NULL)

{

printf("list empty!!!\n");

}

else

{

while(ptr!=NULL)

{

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

ptr=ptr->next;

}

}

printf("\n");

}

void sort()

{

struct node \*temp,\*ptr,\*ptr2;

int a;

temp=head;

ptr=head;

for(temp=head;temp!=NULL;temp=temp->next)

{

for(ptr=temp;ptr!=NULL;ptr=ptr->next)

{

if(ptr->data<temp->data)

{

a=temp->data;

temp->data=ptr->data;

ptr->data=a;

}

}

}

}

void reverse()

{

struct node \*prev=NULL,\*current=head, \*next=NULL;

while(current!=NULL)

{

next=current->next;

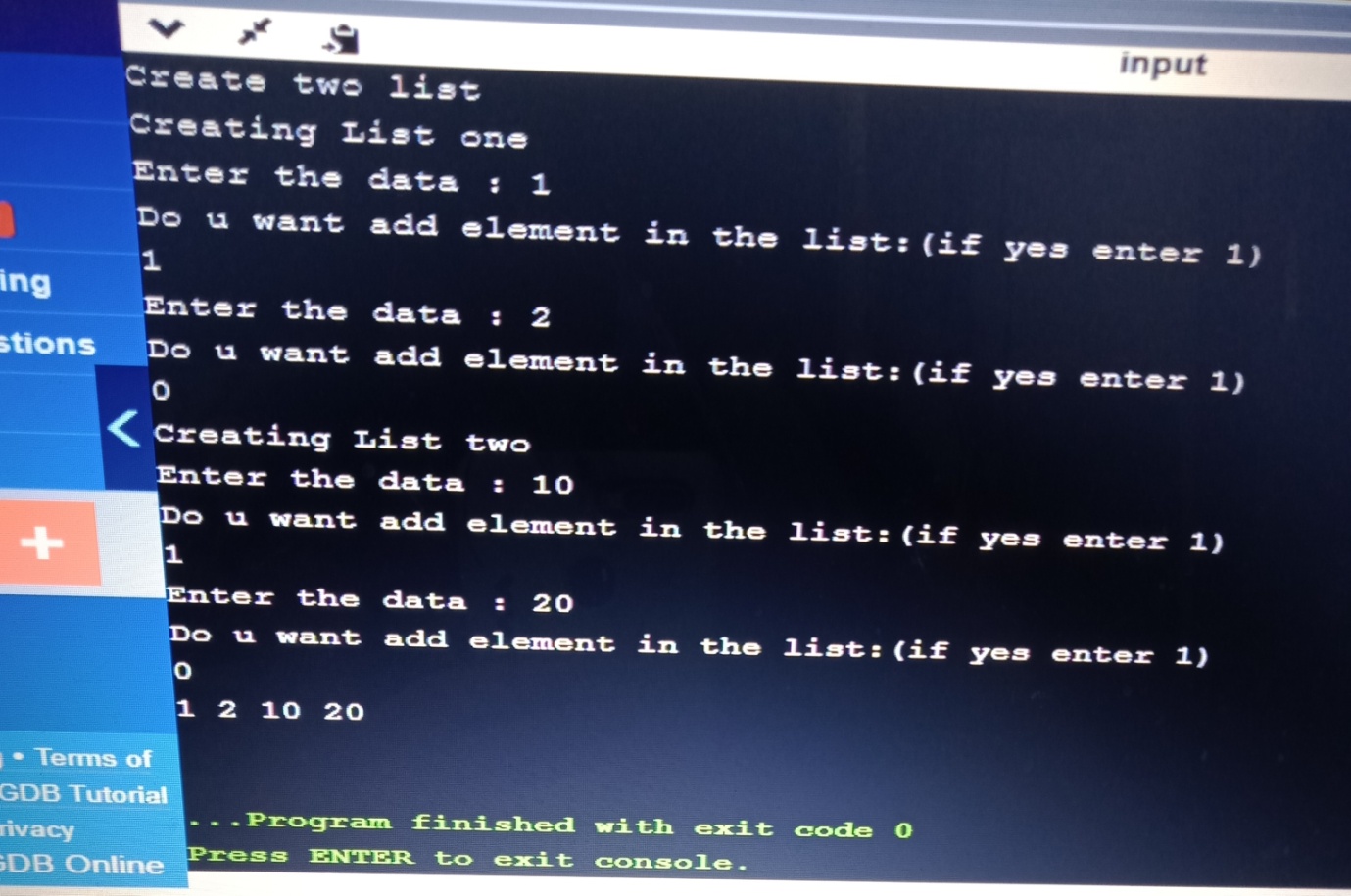
current->next=prev;

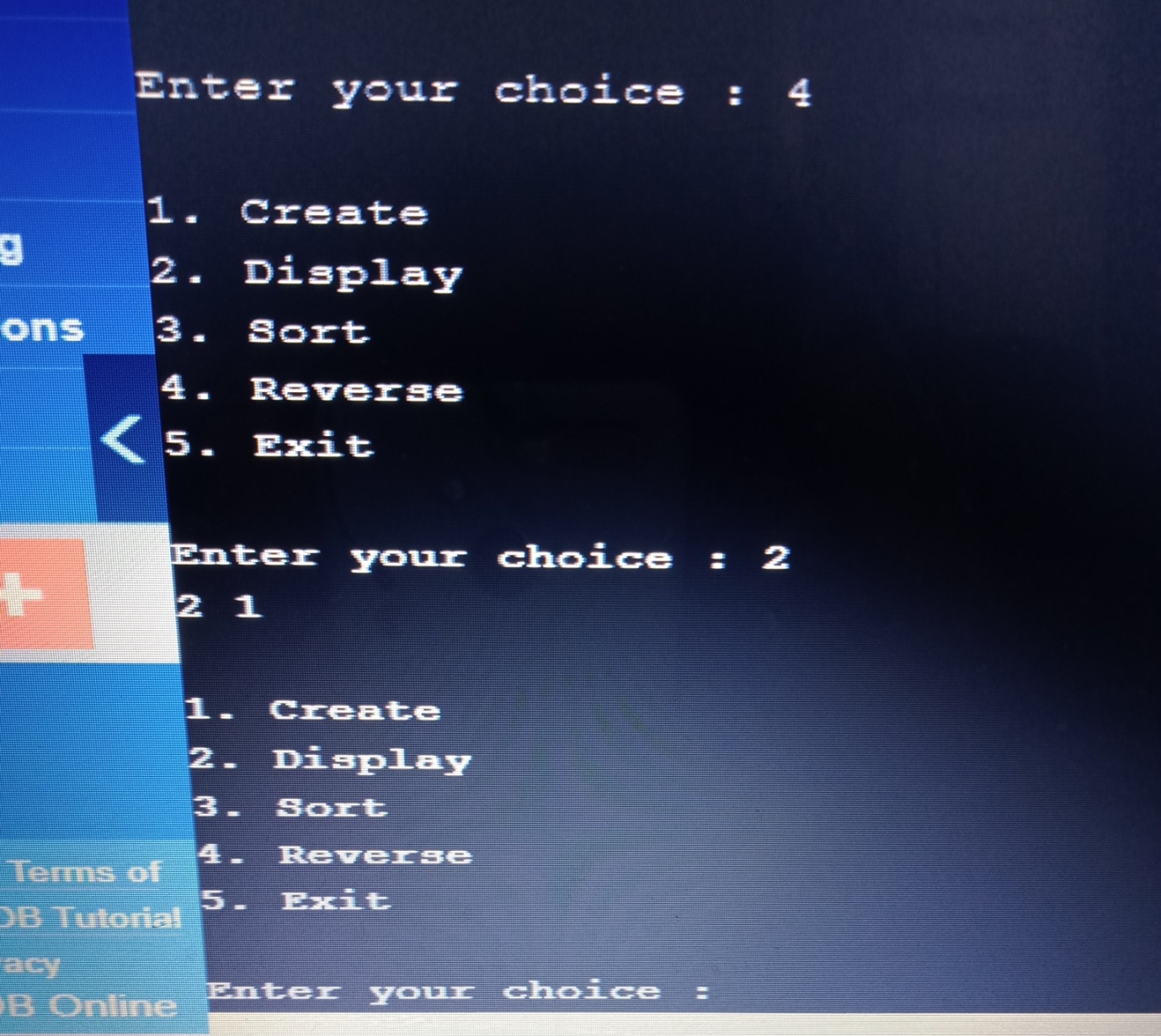
prev=current;

current=next;

}

head=prev;

}



**LAB-8**

WAP to implement Stack & Queues using Linked Representation

#include<stdio.h>

#include<stdlib.h>

void push();

void pop();

void display();

struct node

{

int data;

struct node \*next;

};

struct node \*top=NULL;

int main(int argc, char \*\*argv)

{

int choice;

do

{

printf("\n1. Push \n2. Display \n3. Pop\n4.Exit\n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1: push(); break;

case 2: display();break;

case 3: pop(); break;

case 4:exit(0);break;

default: exit(0);

}

}while(choice!=4);

}

void push()

{

int item;

struct node \*newnode;

printf("Enter the element\n");

scanf("%d",&item);

newnode=(struct node\*)malloc(sizeof(struct node));

newnode->data=item;

newnode->next=NULL;

if(top==NULL)

top=newnode;

else

newnode->next=top;

top=newnode;

}

void pop()

{

if(top==NULL)

printf("stack is empty");

else

{

printf("element removed is %d:", top->data);

top=top->next;

}

}

void display()

{

struct node \*temp;

temp=top;

if(top==NULL)

printf("Stack is empty");

while(temp!=NULL)

{

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

temp=temp->next;

}

}

#include<stdio.h>

#include <stdlib.h>

struct node

{

int data;

struct node \*next;

};

void insert();

void display();

void del();

struct node \*rear=NULL, \*front =NULL;

int main(int argc, char \*\*argv)

{

int choice;

do

{

printf("\n1. Create \n2. Display \n3. Delete \n4. Exit \n");

printf("\nEnter your choice : ");

scanf("%d",&choice);

switch(choice)

{

case 1: insert(); break;

case 2: display();break;

case 3: del(); break;

case 4: exit(0);

default:exit(0);

}

}while(choice!=4);

}

void insert()

{

struct node \*newnode;

newnode=(struct node \*) malloc(sizeof(struct node));

printf("Enter the element:\n");

scanf("%d",&newnode->data);

newnode->next=NULL;

if(rear==NULL)

{

rear=newnode;

front=newnode;

}

else

{

rear->next=newnode;

rear=newnode;

}

}

void del()

{

if(front==NULL)

{

printf("Queue is empty\n");return;

}

else

{

printf("Deleted ele is %d",front->data);

if(front==rear)

{

printf("Queue is empty\n");

front=NULL; rear=NULL;

}

else

front=front->next;

}

}

void display()

{

struct node \*temp;

if(front ==NULL)

{

printf("Queue is empty");

return;

}

temp=front;

while (temp !=NULL)

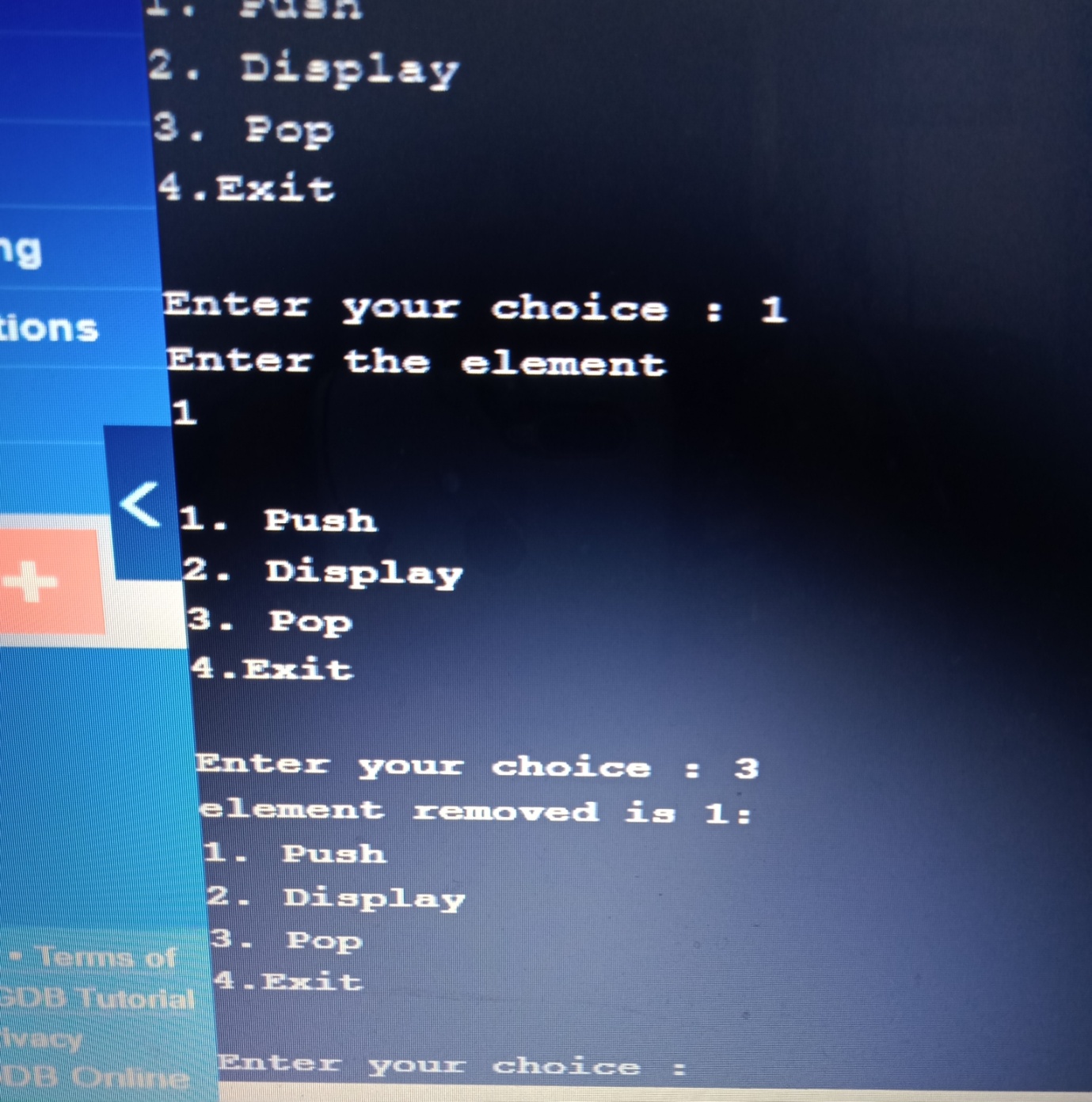
{

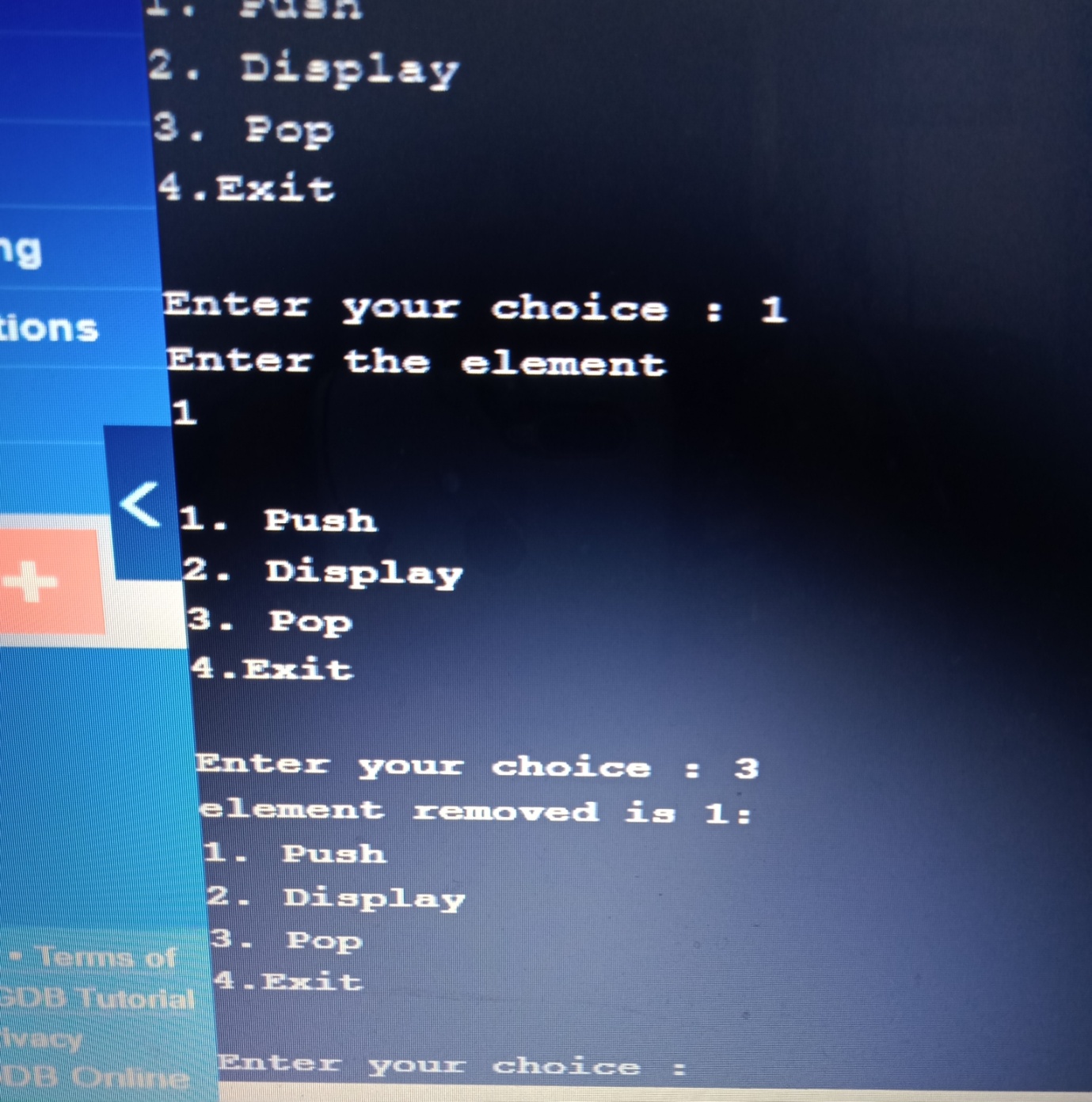
printf("%d ",temp->data);

temp=temp->next;

}

}







**LAB-9**

WAP 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 \*next;

struct node \*prev;

};

struct node \*head=NULL;

void insert\_beg()

{

struct node \*new\_node;

new\_node=(struct node\*)malloc(sizeof(struct node));

printf("Enter the item\n");

scanf("%d",&new\_node->data);

new\_node->next=NULL;

new\_node->prev=NULL;

if(head==NULL)

{

head=new\_node;

}

else

{

new\_node->next=head;

head->prev=new\_node;

head=new\_node;

}

}

void insert\_end()

{

struct node \*new\_node,\*temp;

new\_node=(struct node\*)malloc(sizeof(struct node));

printf("Enter the item\n");

scanf("%d",&new\_node->data);

new\_node->next=NULL;

new\_node->prev=NULL;

if(head==NULL)

{

head=new\_node;

}

else

{

temp=head;

while(temp->next!=NULL)

temp=temp->next;

temp->next=new\_node;

new\_node->prev=temp;

}

}

void insert\_between()

{

int listele;

struct node \*new\_node,\*temp;

printf("Enter the element in the list\n");

scanf("%d",&listele);

new\_node=(struct node\*)malloc(sizeof(struct node));

printf("Enter the new node data\n");

scanf("%d",&new\_node->data);

new\_node->next=NULL;

new\_node->prev=NULL;

if(head==NULL)

{

printf("Empty list\n"); return;

}

temp=head;

while(temp->data!=listele)

{

temp=temp->next;

if(temp==NULL)

{

printf("Element is not in the list");

return;

}

}

if (temp->next==NULL)

{

new\_node->next=temp->next;// becomes null

temp->next=new\_node;

new\_node->prev=temp;

return;

}

new\_node->next=temp->next;// becomes null

temp->next=new\_node;

new\_node->prev=temp;

new\_node->next->prev=new\_node;

}

void del()

{

struct node \*temp;

int ele;

if(head==NULL)

{

printf("Empty List \n");

return;

}

printf("Enter the element to be deleted\n");

scanf("%d",&ele);

temp=head;

while(temp->data!=ele)

{

temp=temp->next;

if(temp==NULL)

{

printf("Element is not in the list\n");

break;

}

}

if(temp==head)

{

head=head->next;

}

else if(temp->next==NULL)

{

temp=temp->prev;

temp->next=NULL;

}

else

{

temp->prev->next=temp->next;

temp->next->prev=temp->prev;

}

}

void display()

{

struct node \*temp;

temp=head;

while(temp!=NULL)

{

printf("%d\t",temp->data);

temp=temp->next;

}

printf("\n");

}

int main()

{

int choice;

while(1)

{

printf(" 1. Insert at the beg \n");

printf(" 2. Insert at the end \n");

printf(" 3. Insert after a given node\n");

printf(" 4. Delete \n");

printf(" 5. Display\n");

printf(" 6. Exit\n");

printf("Enter your choice\n");

scanf("%d",&choice);

switch(choice)

{

case 1: insert\_beg(); break;

case 2: insert\_end();break;

case 3:

insert\_between();break;

case 4: del(); break;

case 5: display(); break;

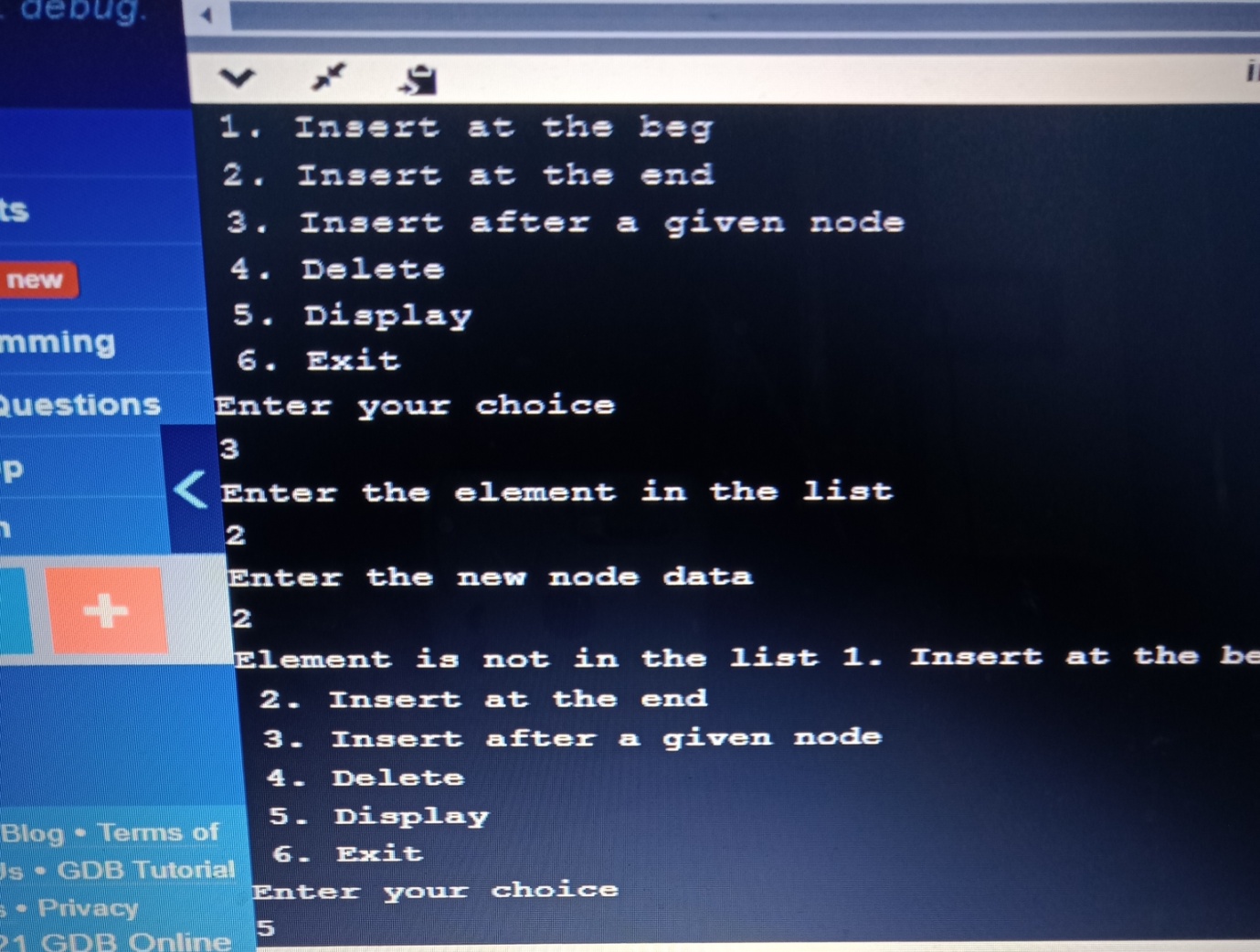
case 6: exit(0);

}

}

return 0;

}



**LAB-10**

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<math.h>

#include<string.h>

#include<stdlib.h>

struct node

{

int data;

struct node \*left;

struct node \*right;

};

typedef struct node \*NODE;

NODE getnode(int data)

{

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

x->data=data;

x->right=NULL;

x->left=NULL;

return x;

}

NODE insert(NODE root,int info)

{

if(root==NULL)

{

root=getnode(info);

return root;

}

else if(info<=root->data)

{

root->left=insert(root->left,info);

}

else

{

root->right=insert(root->right,info);

}

return root;

}

void preorder(NODE root)

{

if(root==NULL)

return;

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

preorder(root->left);

preorder(root->right);

}

void inorder(NODE root)

{

if(root==NULL)

return;

inorder(root->left);

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

inorder(root->right);

}

void postorder(NODE root)

{

if(root==NULL)

return;

postorder(root->left);

postorder(root->right);

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

}

NODE findmin(NODE root)

{

if(root==NULL)

{

return NULL;

}

else if(root->left==NULL)

{

return root;

}

return findmin(root->left);

}

NODE delete\_node(NODE root,int info)

{

if(root==NULL)

{

return root;

}

else if(info<root->data)

{

root->left=delete\_node(root->left,info);

}

else if(info>root->data)

{

root->right=delete\_node(root->right,info);

}

else

{

if(root->left==NULL&&root->right==NULL)

{

free(root);

root=NULL;

return root;

}

else if(root->left==NULL)

{

NODE temp=root;

root=root->left;

free(temp);

return root;

}

else if(root->right==NULL)

{

NODE temp=root;

root=root->right;

free(temp);

return root;

}

else

{

NODE temp=findmin(root->right);

root->data=temp->data;

root->right=delete\_node(root->right,temp->data);

return root;

}

}

}

void display(NODE root,int i)

{

if(root==NULL)

return;

display(root->right,i+1);

for(int j=1;j<=i;j++)

printf(" ");

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

display(root->left,i+1);

}

int main()

{

NODE root=NULL;

int data,option;

do{

printf("1:Insert\n");

printf("2:Delete\n");

printf("3:Preorder\n");

printf("4:PostOrder\n");

printf("5:Inorder\n");

printf("6:Display\n");

printf("7:Exit\n");

printf("Enter Your Choice\n");

scanf("%d",&option);

switch(option)

{

case 1:

printf("Enter The Data To Be Inserted\n");

scanf("%d",&data);

root=insert(root,data);

break;

case 2:

printf("Enter the Data To Be Deleted\n");

scanf("%d",&data);

root=delete\_node(root,data);

break;

case 3:

preorder(root);

printf("\n");

break;

case 4:

postorder(root);

printf("\n");

break;

case 5:

inorder(root);

printf("\n");

break;

case 6:

display(root,1);

break;

}

}while(option!=7);

}

