

B.M.S. College of Engineering
(Autonomous Institution affiliated to VTU, Belagavi)

Department of Computer Science and Engineering



ACADEMIC YEAR: 2020-2021

NAME: AJITH MS

USN: 1BM19CS010

CLASS: CSE-3

SUBJECT: DATA STRUCTURES LAB RECORD

LAB-1

1.

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

```
int top = -1;
void push(int stack[], int ele);
int pop(int stack[]);
void display(int stack[]);
```

```
int main()
{
    int stack[5];
    int i, choice, ele;

    do
    {
        printf("---MENU---\n");
        printf("1. Push\n");
        printf("2. Pop\n");
        printf("3. Display\n");
        printf("4. Exit\n");
        printf("Enter your
choice!\n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1:
                printf("Enter the element
that you want to Push :\n");
```

```

        scanf("%d", &ele);
        push(stack, ele);
        break;

        case 2:
            ele = pop(stack);
            if (ele == -1)
                printf("Stack
Underflow\n");
            else
                printf("The Poped
element is : %d\n", ele);
            break;
        case 3: display(stack);

                break;
        case 4:
            printf("EXITING....\n");
            break;
        default: printf("Invalid
choice!\n");
    }
}
while(choice != 4);
return 0;
}

void push(int stack[], int ele)
{
    if (top==4)
    {
        printf("Stack overflow");
    }
}

```

```

        else
        {
            top++;
            stack[top]=ele;
        }
    }

int pop(int stack[])
{
    int popele;
    if(top==-1)

        return -1;

    else
    {
        popele=stack[top];
        top--;
        return (popele);
    }
}

void display(int stack[])
{
    int i;
    printf("The stack elements\n");
    for(i=top;i>=0;i--)
    {

        printf("%d\n",stack[i]);
    }
}

```

```
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
1
Enter the element that you want to Push :
12
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
1
Enter the element that you want to Push :
13
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
1
Enter the element that you want to Push :
14
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
```

```
4. Exit
Enter your choice!
3
The stack elements
14
13
12
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
2
The Poped element is : 14
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
2
The Poped element is : 13
---MENU---
1. Push
2. Pop
3. Display
4. Exit
Enter your choice!
3
The stack elements
12
---MENU---
```

LAB-2

2.

```

#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);
        case '^': return (3);
        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 '/':
            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;

}

```

```
Enter the infix expression :a+(b*c)-d
Expression : a+(b*c)-d
Postfix: abc*+d-

...Program finished with exit code 0
Press ENTER to exit console.
```

LAB-3

3.

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

```
#include<stdlib.h>
#define maxsize 5
void enqueue(int *Q,int
*front, int *rear)
{
    int ele;
    if(*rear>=maxsize-1)
    {
        printf("Queue is
full.\n");
        return;
    }
    if(*front==-1)
    {
        (*front)++;
    }
}
```

```

        (*rear)++;
        printf("Enter the
element to be inserted");
        scanf("%d",&ele);
        *(Q+*rear)=ele;
    }

```

```

void display(int *Q,int
*front,int *rear)
{
    if(*front== -
1&&*rear== -1)
        printf("Queue is
empty!!!!.\n");
    else
    {
        printf("Elements
in Queue are:\n");
        for(int
i=*front;i<=*rear;i++)
        {
            printf("%d
",*(Q+i));
        }
        printf("\n");
    }
}

```

```

void dequeue(int *Q,int
*front, int *rear)
{
    int ele;

```

```

        if(*front== -
1&&*rear== -1)
        {
            printf("Queue is
empty!!!!\n");
            return;
        }
        else if(*front==*rear)
        {
            ele=*(Q+*front);
            *front=-1;
            *rear=-1;
        }
        else
        {
            ele=*(Q+*front);
            (*front)++;
        }
        printf("Deleted
Element are: %d\n",ele);
    }

```

```

void main()
{
    int front1=-1,rear1=-1;
    int queue1[maxsize];
    int choice;
    printf("1.
Enqueue\n");
    printf("2.
Deque\n");
    printf("3.
Display\n");
}

```

```
        printf("4. Exit\n");

do
{
    printf("Enter your
choice");
    scanf("%d",&choice);

    switch(choice)
    {
        case 1:
enqueue(queue1,&front1,&rear1);
        break;
        case 2:
dequeue(queue1,&front1,&rear1);
        break;
        case 3:
display(queue1,&front1,&rear1);
        break;
        case 4:exit(0);
        break;

default:printf("Please
input correct choice\n");
        break;

    }
}while(choice!=4);

}
```

```
main.c
1 int ele;
2 if( rear ==maxsize )
3 {
4     printf("Queue is full.\n");
5     return;
6 }
7
8 Enter the element to be inserted11
9 Enter your choice1
10 Enter the element to be inserted12
11 Enter your choice1
12 Enter the element to be inserted13
13 Enter your choice1
14 Enter the element to be inserted14
15 Enter your choice1
16 Enter the element to be inserted15
17 Enter your choice1
18 Queue is full.
19 Enter your choice3
20 Elements in Queue are:
21 11 12 13 14 15
22 Enter your choice2
23 Deleted Element are: 11
24 Enter your choice2
25 Deleted Element are: 12
26 Enter your choice2
27 Deleted Element are: 13
28 Enter your choice2
29 Deleted Element are: 14
30 Enter your choice2
31 Deleted Element are: 15
32 Enter your choice2
33 Queue is empty!!!!
34 Enter your choice
```

LAB-4

4.

```
#inc
lude
<std
io.h
>
```

```
#include<stdlib.h>
#define QUE_SIZE 5
int item, front=0, rear=-1,
q[QUE_SIZE],count=0;
```

```
void insertrear()
{
    if(count == QUE_SIZE)
    {
        printf("QUEUE OVERFLOW \n");
        return ;
    }

    rear= (rear+1)%QUE_SIZE;
    q[rear]= item;
    count++;
}
```

```
int deletefront()
{
    if(count==0)
        {return -1;
        }
    item=q[front];
    front=(front+1)%QUE_SIZE;
    count=count-1;
    return item;
}
```

```
void display()
{
    int i,f;
    if(count==0)
    {
        printf("QUEUE IS EMPTY \n");
        return;
    }
    f=front;
```



```

        printf("CONTENTS OF QUEUE \n");
        for(i=1; i<=count; i++)
        {
            printf("%d \n", q[f]);
            f=(f+1)%QUE_SIZE;
        }
    }

```

```

void main()
{
    int choice ;
    for(;;)
    {
        printf("ENTER 1.insert rear  
2.delete front 3.DISPLAY \n");
        printf("ENTER CHOICE \n");
        scanf("%d", &choice);
        switch(choice)
        {
            case 1: printf("ENTER  
ITEM TO BE ENTERED \n");
                    scanf("%d",
&item);
                    insertrear();
                    break;
            case 2: item=
deletefront();
                    if(item == -1)

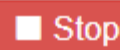
printf("QUEUE EMPTY \n");
                    else

```

```

                                printf("ITEM
DELETED IS %d", item);
                                break;
        case 3: display();
                break;
        default: exit(0);
    }
}
}

```



main.c

```
1 #include<stdio.h>
2 #include<stdlib.h>
3 #define QUE_SIZE 5
4 int item, front=0, rear=-1, q[QUE_
5 void insertrear()
```



```
ENTER 1.insert rear 2.delete front 3.DIS
ENTER CHOICE
1
ENTER ITEM TO BE ENTERED
55
ENTER 1.insert rear 2.delete front 3.DIS
ENTER CHOICE
1
ENTER ITEM TO BE ENTERED
66
ENTER 1.insert rear 2.delete front 3.DIS
ENTER CHOICE
1
ENTER ITEM TO BE ENTERED
77
ENTER 1.insert rear 2.delete front 3.DIS
ENTER CHOICE
2
ITEM DELETED IS 55ENTER 1.insert rear 2.d
ENTER CHOICE
3
CONTENTS OF QUEUE
66
77
ENTER 1.insert rear 2.delete front 3.DIS
ENTER CHOICE
```

LAB-5

5.

```
#i  
nc  
lu  
de  
<s  
td  
io  
.h  
>
```

```
#include <stdlib.h>  
#include <string.h>  
struct node  
{  
    int sem;  
    char name[100];  
    char usn[100];  
    struct node *next;  
};  
struct node *head = NULL;  
int counter = 0;
```

```
void  
Insertst ()  
{  
    struct node *newnode;  
    int s;  
    char n[100], u[100];
```

```

    printf ("\t ----Enter the name-----
: ");
    scanf ("%s", n);
    printf ("\t ----Enter the semester--
--- : ");
    scanf ("%d", &s);
    printf ("\t ----Enter the usn---- :
");
    scanf ("%s", u);
    newnode = (struct node *) malloc
(sizeof (struct node));
    newnode->sem = s;
    strcpy (newnode->name, n);
    strcpy (newnode->usn, u);
    if (head == NULL)
        printf (">>First node created\n");
    newnode->next = head;
    head = newnode;
    counter++;
    printf ("Node created!\n");
}

```

```

void
Insertany (int p)
{
    struct node *newnode;
    int s;
    char n[100], u[100];
    printf ("\t -Enter the name- : ");
    scanf ("%s", n);
    printf ("\t -Enter the semester- :
");
    scanf ("%d", &s);

```

```

printf ("\t -Enter the usn-  : ");
scanf ("%s", u);
newnode = (struct node *) malloc
(sizeof (struct node));
newnode->sem = s;
strcpy (newnode->name, n);
strcpy (newnode->usn, u);
if (p == 1)
{
    printf ("Node of linked list is
inserted in the first position\n");
    newnode->next = head;
    head = newnode;
    counter++;
}
else if (head == NULL && p > 1)
{
    printf ("currently empty!!!\n");
    return;
}
else if (p > (counter + 1))
{
    printf
    ("Not possible since number of pre-
existing nodes in list is
insufficient!\n");
    return;
}
else
{
    struct node *temp1;
    struct node *temp2;
    int count = 1;

```

```

        temp1 = head;
        while (count < (p - 1))
        {
            temp1 = temp1->next;
            count++;
        }
        temp2 = temp1->next;
        temp1->next = newnode;
        newnode->next = temp2;
        counter++;
        printf ("Node inserted at %d
position in linked list\n", p);
    }
}

```

```

void
Insertend ()
{
    struct node *newnode;
    struct node *temp;
    int s;
    char n[100], u[100];
    printf ("Enter the name- : ");
    scanf ("%s", n);
    printf ("Enter the semester- : ");
    scanf ("%d", &s);
    printf ("Enter the usn- : ");
    scanf ("%s", u);
    newnode = (struct node *) malloc
(sizeof (struct node));
    newnode->sem = s;
    strcpy (newnode->name, n);
    strcpy (newnode->usn, u);
}

```

```

    if (head == NULL)
    {
        newnode->next = NULL;
        head = newnode;
        printf(">Very first node
created\n");
        counter++;
    }
    else
    {
        temp = head;
        while (temp->next != NULL)
        {
            temp = temp->next;
        }
        temp->next = newnode;
        newnode->next = NULL;
        counter++;
        printf ("Node created!\n");
    }
}

```

```

void
display ()
{
    struct node *ptr;
    ptr = head;
    int i = 1;

    if (ptr == NULL)
    {
        printf ("Linked list is
empty!!\n");
    }
}

```



```

    }
else
{
    while (ptr != NULL)
    {
        printf ("NODE %d\n", i);
        printf ("Name: %s\n", ptr->name);
        printf ("USN: %s\n", ptr->usn);
        printf ("Sem: %d\n", ptr->sem);
        printf ("-----\n");
        i++;
        ptr = ptr->next;
    }
}

}
int
main ()
{
    int choice, pos;
    do
    {
        printf ("\n");
        printf
        ("\n1. Insert node at starting..
\n2. Insert node anywhere... \n3.
Insert at the end of list.\n4. <
Display list >\n5. Exit!!!\n");
        printf ("\n\t Enter your
choice() : ");
        scanf ("%d", &choice);
        if (choice == 5)
            break;
    }
}

```

```

        switch (choice)
        {
        case 1:
            Insertst ();
            break;

        case 2:
            printf ("Enter in which position
you want to enter your node\n");
            scanf ("%d", &pos);
            Insertany (pos);
            break;

        case 3:
            Insertend ();
            break;

        case 4:
            display ();
            break;

        default:
            printf ("invalid!\n");
            break;
        }
    }
    while (choice != 5);
    return 0;
}

```



DS-LAB-3A1/linkedList.c at maste



Online C Compiler - onlin



onlinegdb.com/online_c_compiler



Apps



Gmail



YouTube



Maps



News



main.c

162

163

printf ("\n);

printf

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

```
Enter your choice() : 1
```

```
----Enter the name----- : kumar
```

```
----Enter the semester----- : 4
```

```
----Enter the usn---- : 1bm19cs414
```

```
>>First node created
```

```
Node created!
```

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

```
Enter your choice() : 1
```

```
----Enter the name----- : rahul
```

```
----Enter the semester----- : 5
```

```
----Enter the usn---- : 1bm19cs345
```

```
Node created!
```

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
```



DS-LAB-3A1/linkedlist.c at maste



Online C Compiler - onlin



onlinegdb.com/online_c_compiler



Apps



Gmail



YouTube



Maps



News



main.c

162

163

printf ("\n);

printf

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

```
Enter your choice() : 2
```

```
Enter in which position you want to enter yo
```

```
2
```

```
-Enter the name- : ramesh
```

```
-Enter the semester- : 3
```

```
-Enter the usn- : 1bm19cs643
```

```
Node inserted at 2 position in linked list
```

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

```
Enter your choice() : 4
```

```
NODE 1
```

```
Name: rahul
```

```
USN: 1bm19cs345
```

```
Sem: 5
```

```
-----
```

```
NODE 2
```

```
Name: ramesh
```



DS-LAB-3A1/linkedList.c at maste



Online C Compiler - onlin



onlinegdb.com/online_c_compiler



Apps



Gmail



YouTube



Maps



News



main.c

162

163

printf ("\n);

printf

```
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

Enter your choice() : 4

NODE 1

Name: rahul

USN: 1bm19cs345

Sem: 5

NODE 2

Name: ramesh

USN: 1bm19cs643

Sem: 3

NODE 3

Name: kumar

USN: 1bm19cs414

Sem: 4

```
1. Insert node at starting..
2. Insert node anywhere...
3. Insert at the end of list.
4. < Display list >
5. Exit!!!
```

Enter your choice() :

LAB-6

6.

```
#i  
nc  
lu  
de  
<s  
td  
io  
.h  
>
```

```
#include <stdlib.h>  
#include <string.h>  
void create();  
void display();  
void insertpos(int);  
void insert_beg();  
void delete();  
void delpos(int);  
void del_beg();  
struct node  
{  
    int sem,usn;  
    char name[20];  
    struct node *next;  
};  
  
struct node *head=NULL;  
int count=0;  
int main(int argc, char **argv)  
{
```

```

    int choice,ele,a;

    do
    {
        printf("\n1.Insert at the end
\n2.Insert at the beginning \n3.
Insert at a position \n4.Delete at the
end \n5.Delete at the beginning
\n6.Delete at a position
\n7.Display\n8.Exit");
        printf("\nEnter your choice : ");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1: create(); break;
            case 2:insert_beg();
            break;
            case 3: printf("Enter the
position to be inserted\n");
                    scanf("%d",&ele);
                    insertpos(ele);
                    break;
            case 4:delete();break;
            case 5:del_beg();break;
            case 6:printf("enter the
position");
                    scanf("%d",&a);
                    delpos(a);

            break;
            case 7:display();
            break;

```

```

        case 8:exit(0);
    }

    }while(choice!=8);
}

void create()
{
    struct node *newnode,*temp;
    int sem1,usn1;
    char name1[20];
    printf("Enter the name USN
semester of student : ");
    scanf("%s %d
%d",name1,&usn1,&sem1);
    newnode =(struct node *) malloc
(sizeof(struct node));

    strcpy(newnode->name,name1);
    newnode->usn=usn1;
    newnode->sem=sem1;
    if (head==NULL)
    {
        newnode->next=NULL;
        head=newnode;
        printf("Node created\n");
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;

```



```

        }
        temp->next=newnode;
        newnode->next=NULL;
        printf("Node created\n");
        count++;
    }
}

void display()
{
    struct node *ptr=NULL;
    ptr=head;

    if(ptr==NULL)
    {
        printf("Nothing to print\n");
    }
    else
    {
        while(ptr!=NULL)
        {
            printf("%s ",ptr->name);
            printf("%d ",ptr->usn);
            printf("%d ",ptr->sem);
            printf("\n");
            ptr=ptr->next;
        }
    }
}

void insertpos(int p)
{

```

```

    struct node *newnode;
    int sem1,usn1;
    char name1[20];
    if(count+2<p)
        printf("the position exceeds the
number of nodes");
    else if(head==NULL&& p>1)
    {
        printf("node empty enter in
first position");
    }
    else
    {
        printf("Enter the name USN
semester of student : ");
        scanf("%s %d
%d",name1,&usn1,&sem1);
        newnode =(struct node *) malloc
(sizeof(struct node));

        strcpy(newnode->name,name1);
        newnode->usn=usn1;
        newnode->sem=sem1;

        if(p==1)
        {
            printf("inserted at the
beginning\n");
            newnode->next=head;
            head=newnode;
            count++;

```

```

    }

    else
    {

        int i;
        struct node *temp1;
        temp1=head;
        for(i=2;i<p;i++)
        {
            temp1= temp1->next;
        }
        newnode->next=temp1->next;
        temp1->next=newnode;

        printf("Node inserted at %d
position in linked list\n",p);
        count++;
    }
}

```

```

void insert_beg()
{
    struct node *newnode;
    int sem1,usn1;
    char name1[20];
    printf("Enter the name USN
semester of student : ");
    scanf("%s %d
%d",name1,&usn1,&sem1);

```

```

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

        strcpy(newnode->name,name1);
        newnode->usn=usn1;
        newnode->sem=sem1;
        newnode->next=head;
        head=newnode;
        count++;
    }

void delete()
{
    struct node *temp=NULL;
    int sem1,usn1;
    char name1[20];
    if(head==NULL)
        printf("linked list is empty");
    else
    {
        temp=head;
        while(temp->next->next!=NULL)
        {
            temp=temp->next;
        }
        strcpy(name1,temp->next-
>name);
        sem1=temp->next->sem;
        usn1=temp->next->usn;
        printf("the student info
deleted = %s %d %d",name1,usn1,sem1);
        temp->next=NULL;
        count--;
    }
}

```

```
    }  
}
```

```
void del_beg()  
{  
    struct node *temp=NULL;  
    int sem1,usn1;  
    char name1[20];  
    if(head==NULL)  
        printf("linked list is empty");  
    else  
    {  
        strcpy(name1,head->name);  
        sem1=head->sem;  
        usn1=head->usn;  
        printf("the student info  
deleted = %s %d %d",name1,usn1,sem1);  
        temp=head;  
        head=temp->next;  
        free(temp);  
        count--;  
    }  
}  
void delpos(int p)  
{  
    struct node *temp=NULL;  
    int sem1,usn1;  
    char name1[20];  
    if(head==NULL)  
        printf("linked list is empty");  
    else if(count+1<p)
```

```

        printf("the position exceeds the
number of nodes");
    else if(p==1)
    {
        strcpy(name1,head->name);
        sem1=head->sem;
        usn1=head->usn;
        printf("the student info
deleted = %s %d %d",name1,usn1,sem1);
        temp=head;
        head=temp->next;
        free(temp);
        count--;
    }
    else
    {
        int i;
        struct node *temp,*ptr;
        temp=head;
        for(i=2;i<p;i++)
        {
            temp= temp->next;
        }

        strcpy(name1,temp->next-
>name);
        sem1=temp->next->sem;
        usn1=temp->next->usn;
        printf("the student info
deleted = %s %d %d",name1,usn1,sem1);
        ptr=temp->next;
        temp->next=temp->next->next;
        free(ptr);
    }
}

```

```
        count--;  
    }  
}
```

7.Display

8.Exit

Enter your choice : 6

enter the position1

the student info deleted = awe 1234

1.Insert at the end I

2.Insert at the beginning

3. Insert at a position

4.Delete at the end

5.Delete at the beginning

6.Delete at a position

7.Display

8.Exit

Enter your choice : 7

tgf 456 4

wqr 134 3

1.Insert at the end

2.Insert at the beginning

3. Insert at a position

4.Delete at the end

5.Delete at the beginning

6.Delete at a position

7.Display

8.Exit

Enter your choice : 8

- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 4

the student info deleted = rty 3

- 1.Insert at the end
- 2.Insert at the beginning
- 3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 5

the student info deleted = oiu 78

- 1.Insert at the end
- 2.Insert at the beginning
- 3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 6

enter the position1

4

Node inserted at 3 position in

- 1.Insert at the end
- 2.Insert at the beginning
3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 7

oiu 789 7

awe 1234 4

tgf 456 4

wqr 134 3

rty 345 5

- 1.Insert at the end
- 2.Insert at the beginning
3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 4



Search

789

7

- 1.Insert at the end
- 2.Insert at the beginning
3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 3

Enter the position to be inserted
3

Enter the name USN semester
456

4

Node inserted at 3 position

- 1.Insert at the end

1.Insert at the end

2.Insert at the beginning

3. Insert at a position

4.Delete at the end]

5.Delete at the beginning

6.Delete at a position

7.Display

8.Exit

Enter your choice : 1

Enter the name USN semest

345

5

Node created

1.Insert at the end

2.Insert at the beginning

3. Insert at a position

4.Delete at the end

5.Delete at the beginning

6.Delete at a position

7.Display

8.Exit

Enter your choice : 2

```
1.Insert at the end
2.Insert at the beginning
3. Insert at a position
4.Delete at the end
5.Delete at the beginning
6.Delete at a position
7.Display
8.Exit
Enter your choice : 1
Enter the name USN seme
134
3
Node created
```



Search



main.c

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 void create()
```

- 1.Insert at the end
- 2.Insert at the beginning
3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display
- 8.Exit

Enter your choice : 1

Enter the name USN seme

1234

4

Node created

- 1.Insert at the end
- 2.Insert at the beginning
3. Insert at a position
- 4.Delete at the end
- 5.Delete at the beginning
- 6.Delete at a position
- 7.Display

LAB-7

7

```
#in  
clu  
de  
<st  
dli  
b.h  
>
```

```
#include <string.h>  
struct node  
{  
    int sem;  
    struct node *next;  
};  
struct node *head= NULL;  
struct node *head2= NULL;  
int c=0;  
void Insert()  
{  
    struct node *newnode;  
    struct node *temp;  
    int s;  
    printf("Enter integer  : ");  
    scanf("%d",&s);  
    newnode=(struct  
node*)malloc(sizeof(struct node));  
    newnode->sem =s;  
    if (head==NULL)  
    {  
        newnode->next=NULL;
```

```

        head=newnode;
        printf("first node of linked
list created\n");
        c++;
    }
    else
    {
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        newnode->next=NULL;
        c++;
        printf("Node created\n");
    }
}
void Insert2()
{
    struct node *newnode;
    struct node *temp;
    int s,y;
    printf("enter elements to create
list 2\n");
    do
    {
        printf("Enter integer  : \n");
        scanf("%d",&s);
        newnode=(struct
node*)malloc(sizeof(struct node));
        newnode->sem =s;
        if (head2==NULL)

```



```

        {
            newnode->next=NULL;
            head2=newnode;
            printf("first node of linked
list created\n");
            c++;
        }
        else
        {
            temp=head2;
            while(temp->next!=NULL)
            {
                temp=temp->next;
            }
            temp->next=newnode;
            newnode->next=NULL;
            c++;
            printf("Node created\n");
        }
        printf("do u want to continue
adding:0 or 1\n");
        scanf("%d",&y);
    }while(y!=0);
}

```

```

void bubbleSort()
{
    int swapped, i;
    struct node *ptr1;
    struct node *lptr = NULL;

```

```

        if (head == NULL)
            return;

        do
        {
            swapped = 0;
            ptr1 = head;

            while (ptr1->next != lptr)
            {
                if (ptr1->sem > ptr1-
>next->sem)
                {
                    int temp = ptr1->sem;
                    ptr1->sem = ptr1-
>next->sem;
                    ptr1->next->sem =
temp;
                    swapped = 1;
                }
                ptr1 = ptr1->next;
            }
            lptr = ptr1;
        }
        while (swapped);
    }

void reverse()
{
    struct node* prev = NULL;
    struct node* current = head;
    struct node* next = NULL;
    while (current != NULL) {

```

```

        next = current->next;
        current->next = prev;
        prev = current;
        current = next;
    }
    head= prev;
}

void concat()
{
    struct node *ptr;
    if(head==NULL)
    {
        head=head2;
    }
    if(head2==NULL)
    {
        head2=head;
    }
    ptr=head;
    while(ptr->next!=NULL)
        ptr=ptr->next;
    ptr->next=head2;
}

void display1()
{
    struct node *ptr;
    ptr=head;
    int i=1;

    if(ptr==NULL)
    {

```

```

        printf("Linked list is
empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {
            printf(" %d",ptr->sem);
            i++;
            ptr=ptr->next;
        }

    }

}

void display2()
{
    struct node *ptr;
    ptr=head2;
    int i=1;

    if(ptr==NULL)
    {
        printf("Linked list is
empty!\n");
    }
    else
    {
        while(ptr!= NULL)
        {

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

```

```

        printf("\n");
        i++;
        ptr=ptr->next;
    }

}

}

int main()
{
    int choice,pos;
    do
    {

        printf("\n1. Insert node \n2.
sort node\n3. reverse node\n4.concat
2 lists \n5.exit\n");
        printf("\nEnter your choice :
");
        scanf("%d",&choice);
        switch(choice)
        {
            case 1:
                Insert();
                break;

            case 2:
                bubbleSort();
                display1();
                break;

            case 3:

```

```
        reverse();  
        display1();  
        break;  
  
        case 4:  
            Insert2();  
            concat();  
            display1();  
            break;  
  
        case 5:  
            break;  
  
        default:  
            printf("Wrong choice!\n");  
            break;  
    }  
}while(choice!=5);  
return 0;  
}
```

Enter your choice : 1

Enter integer : 6

Node created

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists
- 5.exit

Enter your choice : 1

Enter integer : 12

Node created

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists
- 5.exit

Enter your choice : 2

2 4 6 12

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists
- 5.exit

Enter your choice : 1

Enter integer : 2

first node of linked list created

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists
- 5.exit

Enter your choice : 1

Enter integer : 4

Node created

1. Insert node
2. sort node
3. reverse node
- 4.concat 2 lists
- 5.exit

Enter your choice : 1


```
1
Enter integer :
24
Node created
do u want to continue adding:0 or
1
Enter integer :
26
Node created
do u want to continue adding:0 or
0
12 6 4 2 23 24 26
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 2
2 4 6 12 23 24 26
1. Insert node
2. sort node
3. reverse node
4.concat 2 lists
5.exit

Enter your choice : 
```

Enter integer :

23

first node of linked list created

do u want to continue adding:0 or 1

1

Enter integer :

24

Node created

do u want to continue adding:0 or 1

1

Enter integer :

26

Node created

do u want to continue adding:0 or 1

0

12 6 4 2 23 24 26

1. Insert node

2. sort node

3. reverse node

4.concat 2 lists

5.exit

Enter your choice : 2

2 4 6 12 23 24 26

1. Insert node

2. sort node

3. reverse node

5.exit

Enter your choice : 2

2 4 6 12

1. Insert node

2. sort node

3. reverse node

4.concat 2 lists

5.exit

Enter your choice : 3

12 6 4 2

1. Insert node

2. sort node

3. reverse node

4.concat 2 lists

5.exit

Enter your choice : 4

enter elements to create list 2

Enter integer :

23

first node of linked list created

do u want to continue adding:0 or 1

1

Enter integer :

24

LAB-8

8.1

```
#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()
{
printf("\n--QUEUE IMPLEMENTATION
USING LL--\n");
    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);

}
}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 element
is %d",front->data);
        if(front==rear)
        {
            printf("\nQueue 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;
    }
}

```

8.2

```

#include
<std
io.h
>

```

```

#include<stdlib.h>

```

```

void push();
void pop();
void display();
struct node
{
    int data;
    struct node *next;
};
struct node *top=NULL;

int main()
{

```



```

        int choice;
        printf("\n--STACK IMPLEMENTATION
USING LL--\n");
        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);
            }

        }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 popped 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;
    }
}

```

}

}

```
void pop();
```

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 3

element popped is 15

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 3

element popped is 14

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 3

element popped is 13

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice :



Search

```
4 void push();  
5 void pop();
```

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 1

Enter the element

14

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 1

Enter the element

15

- 1. Push
- 2. Display
- 3. Pop
- 4. Exit

Enter your choice : 2

15 14 13 12

- 1. Push
- 2. Display

```
4 void push();  
5 void pop();
```

--STACK IMPLEMENTATION USING LL--

1. Push
2. Display
3. Pop
4. Exit

Enter your choice : 1

Enter the element

12

1. Push
2. Display
3. Pop
4. Exit

Enter your choice : 1

Enter the element

13

1. Push
2. Display
3. Pop
4. Exit

Enter your choice : 1

37 - 3
3. Delete

4. Exit

Enter your choice : 3

Deleted element is 22

1. Create

2. Display

3. Delete

4. Exit

Enter your choice : 3

Deleted element is 23

1. Create

2. Display

3. Delete

4. Exit

Enter your choice : 3

Deleted element is 24

Queue is empty

1. Create

2. Display

3. Delete

4. Exit

Enter your choice :



Search

Enter your choice : 2

21 22 23 24

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 3

Deleted element is 21

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 3

Deleted element is 22

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 3

Deleted element is 23

1. Create
2. Display
3. Delete



```
36 void insert()  
37 {
```

4. Exit

I

Enter your choice : 1

Enter the element:

23

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 1

Enter the element:

24

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 2

21 22 23 24

1. Create
2. Display
3. Delete
4. Exit


```
37 1
--QUEUE IMPLEMENTATION USING LL--

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 1
Enter the element:
21

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 1
Enter the element:
22

1. Create
2. Display
3. Delete
4. Exit

Enter your choice : 1
```

LAB-9

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

```
#include<stdlib.h>  
struct node  
{  
    int data;  
    struct node *next;  
    struct node *prev;  
};  
struct node *head=NULL;  
  
void insert_left()  
{  
    struct node  
*new_node;  
    new_node=(struct  
node*)malloc(sizeof(stru  
ct 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_right()
{
    struct node
    *new_node,*temp;
    new_node=(struct
node*)malloc(sizeof(stru
ct 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_leftpos()
{
    if(head==NULL)
    {
        printf("Empty
list\n"); return;
    }
    int ele;
    struct node
    *new_node,*temp;
    printf("Enter the
element in the list\n");
    scanf("%d",&ele);
    new_node=(struct
node*)malloc(sizeof(stru
ct node));
    printf("Enter the new
node data\n");
    scanf("%d",&new_node-
>data);
    new_node->next=NULL;
    new_node->prev=NULL;

    temp=head;
    if(temp->data==ele)

```

```

        {
            new_node->
next=head;
            head->
prev=new_node;

            head=new_node;
        }
        else if(temp->next==NULL)
        {
            printf("Element
is not in the list\n");
        }
        else
        {
            while(temp->next->data!=ele)
            {
                temp=temp->next;
                if(temp==NULL)
                {

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

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

```

```
        new_node->next-  
>prev=new_node;  
    }
```

```
    }  
void delete()  
{  
    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");  
            return;  
        }  
    }  
}
```

```

        if(temp==head)
//first node
        {
            head=head->next;
        }
        else if(temp-
>next==NULL) //last
node
        {
            temp=temp-
>prev;
            temp-
>next=NULL;
        }

        else //middle
        {
            temp->prev-
>next=temp->next;
            temp->next-
>prev=temp->prev;
            free(temp);
        }
    }
void display()
{
    if(head==NULL)
    {
        printf("Empty
List \n");
    }
    else

```

```

        {
            struct node
*temp;
            temp=head;
            while(temp!=NULL)
            {

printf("%d\t",temp-
>data);
                temp=temp->next;
            }
            printf("\n");
        }
    }

```

```

int main()
{
    int choice;
    do
    {
        printf(" 1.
Insert at the left \n");
        printf(" 2.
Insert at the left of
the specific node \n");
        printf(" 3.
Insert at the right
\n");
        printf(" 4.
Delete a specific
value\n");
    }
}

```



```

                printf(" 5.
Display\n");
                printf(" 6.
Exit\n");
                printf("Enter
your choice\n");

                scanf("%d",&choice);

                switch(choice)
                {
                        case 1:
insert_left(); break;
                        case 2:
insert_leftpos(); break;
                        case 3:
insert_right(); break;
                        case 4:
delete(); break;
                        case 5:
display(); break;
                        case 6:
exit(0);
                }
                }while(choice!=6);
}

```

Enter your choice

1

Enter the item I

13

1. Insert at the left
2. Insert at the left of the specific node
3. Insert at the right
4. Delete a specific value
5. Display
6. Exit

Enter your choice

2

Enter the element in the list

13

Enter the new node data

19

1. Insert at the left
2. Insert at the left of the specific node
3. Insert at the right
4. Delete a specific value
5. Display
6. Exit

Enter your choice

3

Enter the item

4

LAB-10

10.

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

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

```
struct node
```

```
{
```

```
    int data;
```

```
    struct node* left;
```

```
    struct node* right;
```

```
}*root1;
```

```
struct node *create()
```

```
{
```

```
    struct node *temp;
```

```
    printf("Enter ROOT NODE ELEMENT : ");
```

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

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

```
    temp->left=temp->right=NULL;
```

```
    return temp;
```

```
}
```

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

```
void printPostorder(struct node* node)
{
    if (node == NULL)
```

```
    return;
    printPostorder(node->left);
    printPostorder(node->right);
    printf("%d\t", node->data);
}
```

```
void printInorder(struct node* node)
{
    if (node == NULL)
        return;
    printInorder(node->left);
    printf("%d\t", node->data);
    printInorder(node->right);
}
```

```
void printPreorder(struct node* node)
{
    if (node == NULL)
        return;
    printf("%d\t", node->data);
```

```
    printPreorder(node->left);  
    printPreorder(node->right);  
}
```

```
int main()  
{  
    int choice;  
    struct node* temp;  
    do  
    {  
        printf("\n-----MENU-----\n");  
        printf("1. CREATE\n");  
        printf("2. INSERT\n");  
        printf("3. PREORDER TRAVERSAL\n");  
        printf("4. INORDER TRAVERSAL\n");  
        printf("5. POSTORDER TRAVERSAL\n");  
        printf("6. EXIT\n");  
        printf("Enter your choice correctly : \n");  
        scanf("%d", &choice);  
        switch(choice)  
        {  
            case 1: root1 = create();
```

```

        break;
    case 2: printf("Enter the VALUE you want to
INSERT : ");

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

        scanf("%d",&temp->data);
        insert(root1, temp);
        break;
    case 3: printPreorder(root1);
        break;
    case 4: printInorder(root1);
        break;
    case 5: printPostorder(root1);
        break;
    case 6: printf("EXITING...!!!");
        break;
    default: printf("Incorrect Choice!!\n");
}
}while(choice != 6);
return 0;
}

```




DS-LAB-3A1/LAB-9 at master · ajj x



WhatsApp



Online



onlinegdb.com/online_c_compiler



Apps



Gmail



YouTube



Maps



News



Run



Debug



Stop



Share



Save



{ } Beautify

main.c

```
78 printf("4. INORDER TRAVERSAL\n");
79 printf("5. POSTORDER TRAVERSAL\n");
80 printf("6. EXIT\n");
81 printf("Enter your choice correctly : \n");
82 scanf("%d", &choice);
83 switch(choice)
```



```
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
4
14      33      44      55
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
5
55      44      33      14
-----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :

```



DS-LAB-3A1/LAB-9 at master · ajj x



WhatsApp x



Online



onlinegdb.com/online_c_compiler



Apps



Gmail



YouTube



Maps



News



Run



Debug



Stop



Share



Save



{ } Beautify

main.c

```
78 printf("4. INORDER TRAVERSAL\n");
79 printf("5. POSTORDER TRAVERSAL\n");
80 printf("6. EXIT\n");
81 printf("Enter your choice correctly : \n");
82 scanf("%d", &choice);
83 switch(choice)
```



6. EXIT

Enter your choice correctly :

2

Enter the VALUE you want to INSERT : 55

-----MENU-----

1. CREATE

2. INSERT

3. PREORDER TRAVERSAL

4. INORDER TRAVERSAL

5. POSTORDER TRAVERSAL

6. EXIT

Enter your choice correctly :

3

14 33 44 55

-----MENU-----

1. CREATE

2. INSERT

3. PREORDER TRAVERSAL

4. INORDER TRAVERSAL

5. POSTORDER TRAVERSAL

6. EXIT

Enter your choice correctly :

4

14 33 44 55

-----MENU-----

1. CREATE

DS-LAB-3A1/LAB-9 at master · onlinegdb.com · GitHub

WhatsApp

Online C Compiler - online editor

+

onlinegdb.com/online_c_compiler

Apps Gmail YouTube Maps News

Run Debug Stop Share Save Beautify

Language C

main.c

```
78 printf("4. INORDER TRAVERSAL\n");
79 printf("5. POSTORDER TRAVERSAL\n");
80 printf("6. EXIT\n");
81 printf("Enter your choice correctly : \n");
82 scanf("%d", &choice);
```

input

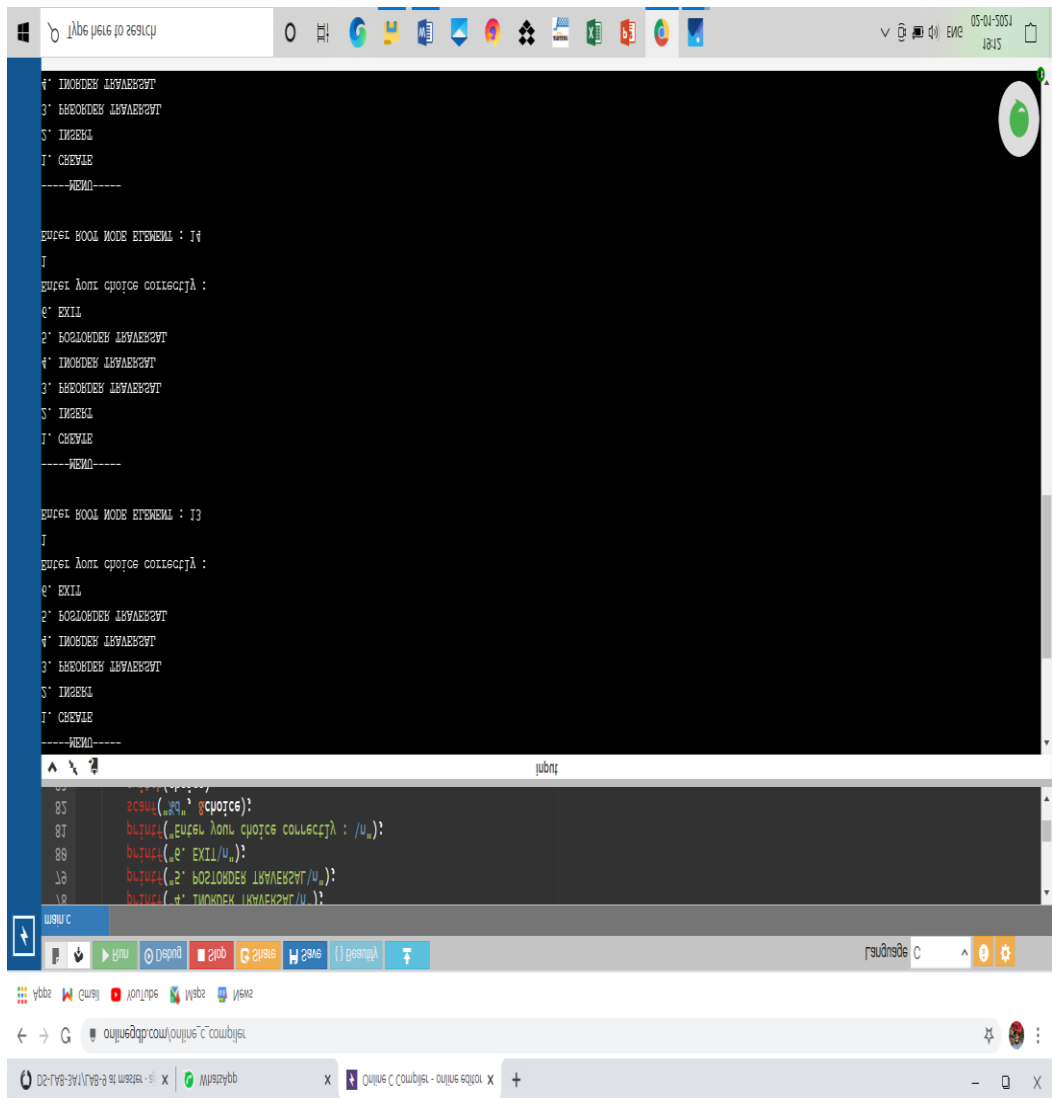
```
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 33

----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
6. EXIT
Enter your choice correctly :
2
Enter the VALUE you want to INSERT : 44

----MENU-----
1. CREATE
2. INSERT
3. PREORDER TRAVERSAL
4. INORDER TRAVERSAL
5. POSTORDER TRAVERSAL
```

Type here to search

19:12 02-01-2021



END...

THANK YOU

AJITH MS

1BM19CS010

CSE 3-A