

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



DATA STRUCTURE LAB RECORD

Submitted by

SHREEHARI KULKARNI (1BM19CS153)

Under the Guidance of

Prof. SHEETAL VA
Assistant Professor, BMSCE

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
Sep-2020 to Jan-2021

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 RECORD carried out by **SHREEHARI KULKARNI(1BM19CS153)** who is the bonafide students 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 Visveswaraiah Technological University, Belgaum during the year 2020-2021. The lab report has been approved as it satisfies the academic requirements in respect of **DATA STRUCTURE LAB RECORD (19CS3PCDST)** work prescribed for the said degree.

Signature of the Guide
Prof. Prof. Sheelal VA
Assistant Professor
BMSCE, Bengaluru

Signature of the HOD
Dr. Umadevi V
Associate Prof.& Head, Dept. of CSE
BMSCE, Bengaluru

External Viva

Name of the Examiner

Signature with date

1. _____

2. _____

INDEX

SL NO	PROGRAM	PAGE NO
1	<p>Write a program to simulate the working of stack using an array with the following:</p> <p>a) Push b) Pop c) Display</p> <p>The program should print appropriate messages for stack overflow, stack underflow</p>	5-10
2	<p>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)</p>	11-15
3	<p>WAP to simulate the working of a queue of integers using an array. Provide the following operations</p> <p>a) Insert b) Delete c) Display</p> <p>The program should print appropriate messages for queue empty and queue overflow conditions</p>	16-20
4	<p>WAP to simulate the working of a circular queue of integers using an array. Provide the following operations.</p> <p>a) Insert b) Delete c) Display</p> <p>The program should print appropriate messages for queue empty and queue overflow conditions</p>	21-26

5	<p>WAP to Implement Singly Linked List with following operations</p> <p>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.</p>	27-41
6	<p>WAP to Implement Singly Linked List with following operations</p> <p>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.</p>	27-41
7	<p>WAP Implement Single Link List with following operations</p> <p>a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists</p>	27-41
8	<p>WAP to implement Stack & Queues using Linked Representation</p>	42-48
9	<p>WAP Implement doubly link list with primitive operations</p> <p>a) Create a doubly linked list. b) Insert a new node to the left of the node.</p> <p>c) Delete the node based on a specific value. c) Display the contents of the list</p>	49-62
10	<p>Write a program</p> <p>a) To construct a binary Search tree.</p> <p>b) To traverse the tree using all the methods i.e., in-order, preorder and post order</p> <p>c) To display the elements in the tree.</p>	63-83

1: Write a program to simulate the working of stack using an array with the following:

a) Push b) Pop c) Display

The program should print appropriate messages for stack overflow, stack underflow

```
#include < stdio.h >
#define STACK_SIZE 5
int top = -1;
int s[10];
int item;
void push()
{
    if (top == STACK_SIZE - 1)
        printf (" stack overflow\n");
    return;
}
top += 1;
s[top] = item;

int pop()
{
    if (top == -1)
        return -1;
    return s[top--];
}

void display()
{
    int i;
    if (top == -1)
        printf (" Stack is empty\n");
    return;
}
```

18

Date _____

```
for(i = top; i >= 0; i--)  
    printf("%d\n", sc[i]);  
  
}  
  
void main()  
{  
    int item_delet;  
    int choice;  
    for(; ;)  
        printf("1: Push 2: Pop 3: display  
               4: exit 1n");  
  
        printf(" Enter the choice1n");  
        switch(choice){  
            case 1:  
                printf("Enter the item to be inserted  
                      1n");  
  
                scanf("%d", &item);  
                push();  
                break;  
        }  
}
```

Date: _____

case 2:

item_deleted = pop()

if (item_deleted == -1)

printf("Stack is empty\n");

else

printf("Deleted item is %d\n", item_deleted)

}

break;

case 3:

display();

break;

default

case 4:

do { exit(0); }

}

OUTPUT:

- 1: Push
- 2: Pop
- 3: display
- 4: exit

Enter your choice:

1

Enter the item to be inserted

25

- 1: Push
- 2: pop
- 3: display
- 4: exit

Enter your choice:

1

Enter the item to be inserted

26

- 1: Push
- 2: pop
- 3: display
- 4: exit

Enter your choice:

1

Enter the item to be inserted

27

- 1: Push
- 2: pop
- 3: display
- 4: exit

OUTPUT:

```
C:\Users\Shreehari.Kulkarni\Documents\BM19CST19\SHREEHARI KULKARNI\LAB1\IMPLEMENTING STACK USING ARRAY.exe"
1:push
2:pop
3:display
4:exit
Enter your choice
2
Stack is empty

1:push
2:pop
3:display
4:exit
Enter your choice
3
Elements of stack are
STACK IS EMPTY NO ITEM CAN BE PRINTED

1:push
2:pop
3:display
4:exit
Enter your choice
4

Process returned 0 (0x0) execution time : 664.350 s
Press any key to continue.
```

```
C:\Users\Shreehari Kulkarni\Documents\IBM19CS153_SHREEHARI KULKARNI_LAB1_IMPLEMENTING_STACK_USING_ARRAY.exe

1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
25

1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
26

1:push
2:pop
3:display
4:exit
Enter your choice
1
Enter the item to be inserted
27

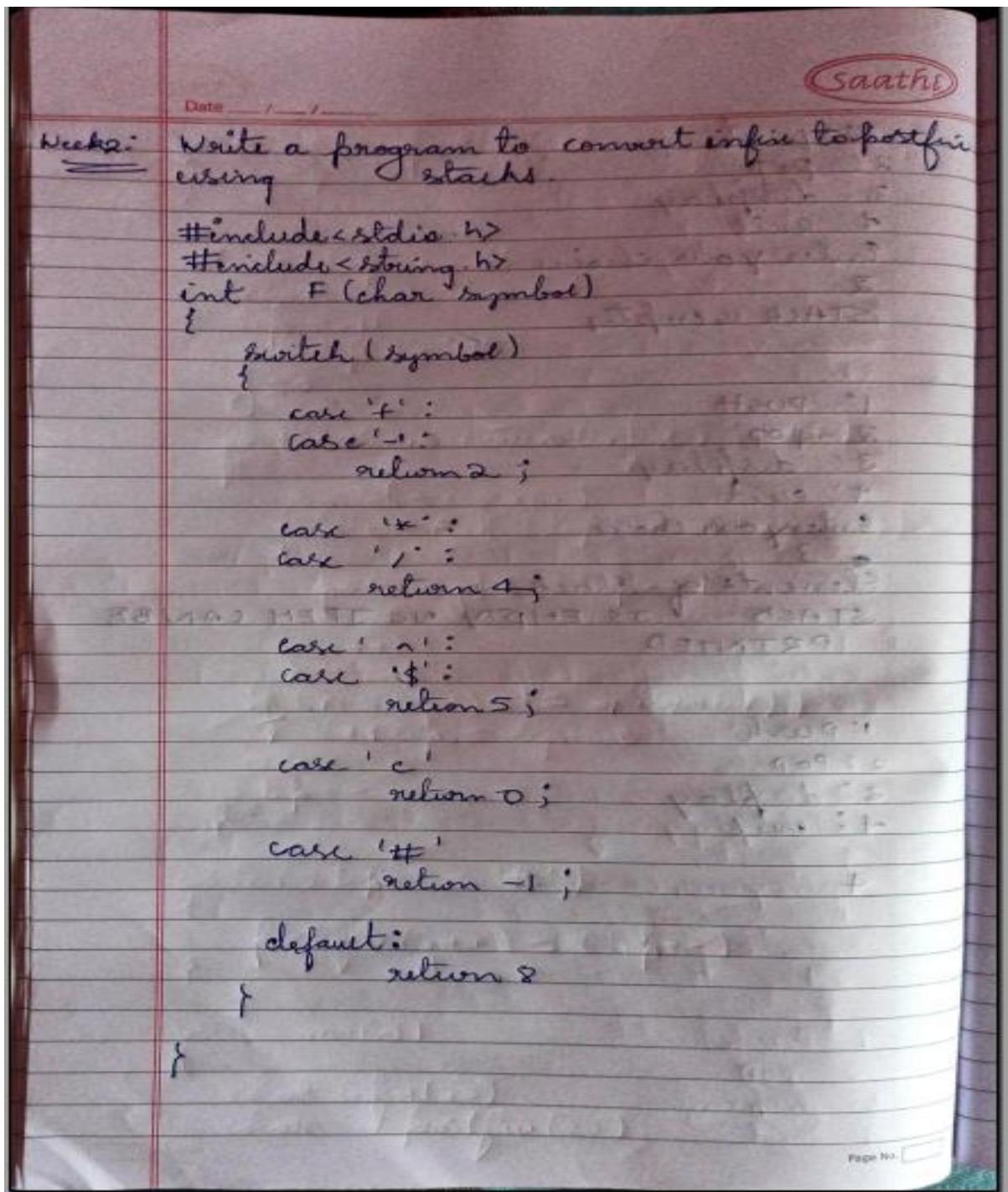
1:push
2:pop
3:display
4:exit
Enter your choice
3
Elements of stack are
27
26
25

1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 27

1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 26

1:push
2:pop
3:display
4:exit
Enter your choice
2
Deleted item is 25
```

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)



Date: _____ / _____ / _____

```
int b( char symbol )
```

```
{ switch (symbol)
```

```
    case '+':
```

```
    case '-':
```

```
        return 1;
```

```
    case '*':
```

```
    case '/':
```

```
        return 3;
```

```
    case '^':
```

```
    case '$':
```

```
        return 6;
```

```
    case 'i':
```

```
        return 9;
```

```
    case ')':
```

```
        return 0;
```

```
    default
```

```
        return 7;
```

Date _____

void infix2postfix(char infix[], char post[])

{

```
int top, i, j;
char s[20], symbol;
top = -1;
s[++top] = '#';
j = 0;
```

```
for (i = 0; i < strlen(infix); i++)
```

{

```
symbol = infix[i];
```

```
while (F(s[top]) > G(symbol))
```

{

```
postfix[j] = s[top--];
```

```
> j++;
```

```
if (F(s[top])) = G(symbol))
```

```
> s[++top] = symbol;
```

```
else
```

```
top--
```

{

```
while (s[top] != '#')
```

{

```
postfix[j++] = s[top--]
```

```
void main()
```

{

```
char infix[20]
```

```
char postfix[20]
```

Date _____

(a) `printf ("Enter the valid expression\n");`
`scanf ("%f", &infix);`
`infix postfin(infix, postfin);`
`printf ("postfin is %n");`
`printf ("%f\n", postfin);`

{

OUTPUT

Enter the valid Expression

$$(A + (B - C) * D)$$

Post fin Expression is

$$ABC - D^* + .$$

OUTPUT:

```
C:\Users\Shrehan Kulkarni\Desktop\infix_postfix_modified.exe
Enter the valid Expression
(a+b)*(c-d)
Valid Expression Continue
Postfix Expression is
ab+cd.*

Process returned 0 (0x0) execution time : 48.132 s
Press any key to continue.
```

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 < math.h >
#include < stdlib.h >
#define max 5
int q[max]
int front = -1, rear = -1
void insert();
int delete();
void display();
int main()
{
    int option, val
    printf("***** Menu *****\n");
    printf(" 1: Insert\n");
    printf(" 2: Delete\n");
    printf(" 3: Display\n");
    printf(" 4: exit\n");
    printf(" Enter the option\n");
    scanf("%d", &option)
    switch(option)
    {
        case 1:
            insert()
            break;
        case 2:
            val = delete();
            if (val != -1)
                printf("The deleted
is %d\n",
```

break;

case 3:

desplay();
break;

case 4:

exit(0);
}

while(option != 5);
return 0;

void insert()

{
int num;
printf("Enter the item to be inserted \n");

scanf("%d", &num);

if(rear == max-1)

{
printf("overflow\n");

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

front = 0;

rear = 0;

q[rear] = num;

{
else

{

}{
near++;

q[near] = num;

```

int delete()
{
    int val;
    if (front == -1 || front > rear)
    {
        printf("UNDERFLOW\n");
        return -1;
    }
    else
    {
        val = q[front];
        front++;
        if (front > rear)
        {
            front = -1;
            rear = -1;
        }
        return val;
    }
}

int display()
{
    if (front == -1 || front > rear)
    {
        printf("Queue is empty\n");
    }
    else
    {
        for (int i = front; i <= rear; i++)
        {
            printf("%d\n", q[i]);
        }
    }
}

```

OUTPUT

1: Insert

2: Delete

3: Display

Enter your option

Enter the number to inserted

2

1: insert

2: Delete

3: Display

4: exit

Enter your option

1

Enter the number to be inserted

6

1: insert

2: Delete

3: display

4: exit

Enter your option

3

2

```
C:\Users\Shreehari Kulkarni\Desktop\ds lab programs\lab5_2.exe
```

```
1:Insert  
2>Delete  
3:display  
4:exit  
Enter your option  
1  
Enter the number to be inserted  
2  
1:Insert  
2>Delete  
3:display  
4:exit  
Enter your option  
1  
Enter the number to be inserted  
6  
1:Insert  
2>Delete  
3:display  
4:exit  
Enter your option  
3  
2  
6  
1:Insert  
2>Delete  
3:display  
4:exit  
Enter your option
```

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

LIFE IS

Saath

I am implementing circular Queue

```
#include <stdio.h>
#define MAX 5
int Queue[MAX];
int front = -1, rear = -1;
void insert(void);
int delete(void);
void display(void);
int main()
{
    int option, val;
    clrscr();
    do
    {
        printf("1: Insert\n");
        printf("2: Delete\n");
        printf("3: Display\n");
        printf("Enter your option\n");
        scanf("%d", &option);
        switch(option)
        {
            case 1:
                insert();
                break;
            case 2:
                val = delete();
                if (Val != -1)
                    printf("Deleted Number is %d\n", Val);
                break;
            case 3:
                display();
                break;
        }
    } while (option != 0);
}
```

Page No. []

Date _____ / _____ / _____

```
while(option != 5)
    return;
```

```
void insert()
```

```
{ int num;
    printf("Enter the number to be added\n");
    scanf("%d", &num);
    if ((front == 0 && rear == MAX - 1))
        printf("overflow\n");
```

```
else if (front == -1 && rear == -1)
{ front = rear = 0;
    queue[rear] = num;
}
```

```
else if (rear == MAX - 1 && front == 0)
```

```
{ rear = 0;
    queue[rear] = num;
}
```

```
else
{
```

```
    rear++;
    queue[rear] = num;
}
```

```
}
```

```

int delete()
{
    int val
    if (front == -1 & rear == -1)
        printf ("In Underflow");
    return -1
}
val = queue(front)
if (front == rear)
    front = rear = -1
else
{
    if (front == Max - 1)
        front = 0
    else
        front++
}
return val
}

```

```

void display()
{
    int i
    printf ("\n");
    if (front == -1 & rear == -1)
        front ("In Queue is empty");
    else
    {
        if (front < rear)

```

Date _____

{

```
for (int i = front; i < max; i++)  
    printf("%d", queue[i])
```

~~front < 0;~~

}

else

{

```
for (int i = front; i < max; i++)  
    printf("%d", queue[i])
```

```
for (int i = 0; i < rear; i++)
```

```
    printf("%d", queue[i])
```

}

}

}

Date _____

OUTPUT

- 1: INSERT
- 2: DELETE
- 3: DISPLAY
- 4: EXIT

Enter your option
1

Enter the number to be inserted
10

- 1: INSERT
- 2: DELETE
- 3: DISPLAY
- 4: EXIT

Enter your option
3

OUTPUT

```
C:\Users\Shreehari Kulkarni\Desktop\ds lab programs\lab5_2.exe"
1:Insert
2>Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
2
1:Insert
2>Delete
3:display
4:exit
Enter your option
1
Enter the number to be inserted
6
1:Insert
2>Delete
3:display
4:exit
Enter your option
3
2
6
1:Insert
2>Delete
3:display
4:exit
Enter your option
```

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.

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.

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

Date _____ / _____ / _____

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

struct Node
{
    int item;
    struct Node *next;
};

Node getNode()
{
    Node n;
    n = (Node)malloc(sizeof(struct node));
    return n;
}
```

```
Node insertFront(Node first, int data)
```

```
{  
    Node new_node;  
    new_node = getNode();  
    new_node->item = data;  
    new_node->next = first;  
    if (first == NULL)  
        return new_node;  
    new_node->next = first;  
    first = new_node;  
    return first;  
}
```

Date _____

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

{

```
Node last;
```

```
Node new_node;
```

```
new_node = getnode();
```

```
new_node->item = data;
```

```
new_node->next = NULL;
```

```
if (first == NULL)
```

```
    return new_node;
```

```
last = first;
```

```
while (last->next != NULL)
```

```
    last = last->next;
```

```
last->next = new_node;
```

```
return first;
```

}

```
Node insert_pos (Node first, int data, int pos)
```

{

```
Node last;
```

```
Node new_node;
```

```
new_node = getnode();
```

```
new_node->item = data;
```

```
if (first == NULL && pos == 1)
```

```
    new_node->next = NULL;
```

```
    return new_node;
```

}

Date _____

```
else if (pos<1)
{
    printf ("Invalid Position\n");
    return first;
}

else if (pos==1)
{
    new_node->next = first;
    first = new_node;
    return first;
}

else
{
    last = first;
    for(int i=1; i< pos-1; i++)
        last = last->next;
    new_node->next = last->next;
    last->next = new_node;
    return first;
}

void display(Node first)
{
    Node temp;
    if (first == NULL)
        printf ("List Empty\n");
    for (temp = first; temp != NULL; temp = temp->next)
        printf ("%d\n", temp->data);
```

Date / /

Node delete front(Node first)

{
Node temp;
if (first == NULL)

printf ("list is empty.\n");
return first;

}
temp = first;
temp = temp -> next;
free(first);
return temp;

Node delete end(Node first)

{
Node temp;
Node prev, cur;
if (first == NULL)

printf ("list empty.\n");
break;

}
cur = first;
while !(cur->next == NULL)

{
prev = cur;
cur = cur->next;

}
prev->next = NULL;
free (cur);
return first;

Date _____ / _____ / _____

```
Node delete_pos(Node first, int pos)
```

```
{ Node prev, cur  
if (first == NULL)
```

```
printf ("list is empty\n");  
return first;
```

```
} if (pos == 0)
```

```
cur = first;  
first = first->next;  
free(cur);  
return first;
```

```
cur = first;  
for (int i = 1; i < pos; i++)
```

```
{ prev = cur;  
cur = cur->next;  
prev->next = cur->next;  
free(cur);  
return first;
```

Date / /

Node reverse (Node first)

```
{ Node cur, prev, temp;
  if (first == NULL)
```

```
  printf ("list is empty\n");
  return first;
```

```
} cur=NULL
```

```
while (first != NULL)
```

```
{ temp = first;
  first = first->next;
  temp->next = cur;
  cur = temp;
```

```
return cur;
```

```
}
```

Node doc

void sort(Node *first)

```
{ int i;
```

```
Node temp, t;
```

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

```
  for (Node j = i->next; j != NULL; j =
```

```
temp-
```

```
  if (i->data > j->data)
```

```
    t = i->data;
```

```
    i->data = j->data;
```

```
    j->data = t;
```

Page No.

Date _____

Node merge sort (Node a, Node b)

{ Node result;

if (b == NULL)

{

result = a;

return result;

}

else if (a == NULL)

{

result = b;

return result;

}

else

{

Node cur = a

while (cur->next != NULL)

cur = cur->next;

^ - a

a = cur->next = b;

^ - b

return cur;

Y

^ .

Date _____ / _____ / _____

```
int main()
```

{

```
    Node head, a, b; =NULL;  
    int data, pos, choice; in;  
    do {  
        printf ("1: Insert front \n");  
        printf ("2: Insert End \n");  
        printf ("3: Insert Pos \n");  
        printf ("4: Delete front \n");  
        printf ("5: Delete End \n");  
        printf ("6: Delete Pos \n");  
        printf ("7: Sort \n");  
        printf ("8: Reverse \n");  
        printf ("9: Concat \n");  
        printf ("Enter Your choice \n");  
        scanf ("%d", &choice);  
        printf ("10: Display \n");  
        printf ("11: Exit \n");  
        switch (choice)  
    }
```

}

case 1:

```
    printf ("Enter the data \n");  
    scanf ("%d", &data);  
    head = insert_front (head, data);  
    break;
```

case 2:

```
    printf ("Enter the data \n");  
    scanf ("%d", &data);  
    head = insert_end (head, data);  
    break;
```

case 3:

```
    printf ("Enter the position \n");  
    scanf ("%d", &pos);
```

Date _____ / _____ / _____

```
printf ("Enter the data [n]\n");
scanf ("%d", &data);
head = insert_pos(head, data, pos);
break;
```

case 4:

```
head = delete_front(head);
break;
```

case 5:

```
head = delete_end(head);
break;
```

case 6:

```
printf ("Enter position [n]\n");
scanf ("%d", &pos);
head = delete_pos(head, pos);
break;
```

case 7:

```
sort(head);
break;
```

case 8:

```
head = reverse(head);
break;
```

case 9:

```
printf ("Enter Number of nodes in list [n]\n");
scanf ("%d", &n);
for(i=1; i<n; i++)
{
    printf ("Enter data [n]\n");
    scanf ("%d", &data);
```

Date / /

```
a = insert_front(a, data);
}
printf ("Enter nodes in linked list 1\n");
scanf ("%d", &n);
for (int i = 0; i < n; i++)
{
    printf ("Enter data\n");
    scanf ("%d", &data);
    b = insert_front(b, data);
}
head = merged_sort(a, b);
break;
case 10:
    display(head);
    break;
} while (option != 1);
```

OUTPUT

```
C:\Users\Shreesh Kukam\OneDrive\Desktop\LABS\CH05 LAB\final_linkedlist.exe'
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
1
Enter the Value To Be Inserted
15
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
1
Enter the Value To Be Inserted
25
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
2
Enter the Value
15
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
2
Enter the Value
20
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
1
25
15
20
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
```

```
C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECTS\LAB\final_linked_list.exe
7:Display
Enter your choice
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
7
15
35
59
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
5
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
7
15
35
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
Enter your choice
```

```
*C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe*
Enter your choice
7
25
15
30
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
9
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
4
15
25
30
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
8
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
```

```
C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\final_linked_list.exe
6>Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
30
25
15
4
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
12
Enter the Number of Nodes in The List
2
Enter the Items In Ascending order for list 1
Enter the item 1 to be inserted at List 1
5
Enter the item 2 to be inserted at List 1
10
Enter the item in ascending order for list 2
Enter the item 1 to be inserted at List 1
2
Enter the item 2 to be inserted at List 1
7
2
5
7
1:Insert at Front
2:Insert at End
3:Insert at Specified Position
4:Delete Front
5:Delete End
6:Delete At Specified Position
7:Display
8:Reverse the list
9:Sort the List
10:Concat
11: Ordered List
12: Merged Sort
13: Exit
Enter your choice
7
30
25
15
4
```

8: WAP to implement Stack & Queues using Linked Representation

Saathi

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

Struct node
{
    int item;
    Node *next;
};

Node getnode()
{
    Node x;
    x = (Node)malloc(sizeof(Struct node));
    return x;
}

Node insert_front(Node first, int data)
{
    Node new_node;
    new_node = getnode();
    new_node->item = data;
    new_node->next = NULL;
    if (first == NULL)
    {
        return new_node;
    }
    new_node->next = first;
    first = new_node;
    return first;
}

Node insert_end(Node first, int data)
{
    Node last;
    Node new_node;
    new_node = getnode();
    new_node->next = NULL;
    if (first == NULL)
```

Date ____ / ____ / ____

```

    return new_node;
}
last = first;
while (last->next != NULL)
{
    last = last->next;
}
last->next = new_node;
return first;
}

Node delete_front (Node first)
{
    Node temp;
    if (first == NULL)
    {
        printf ("List is empty cannot be deleted\n");
        return first;
    }
    temp = first;
    temp = temp->next;
    free (first);
    return temp;
}

Node delete_end (Node first)
{
    Node prev, cur;
    if (first == NULL)
    {
        printf ("List is empty And cannot be
                deleted\n");
        return first;
    }
    cur = first;
    while (cur->next != NULL)
    {
        prev = cur;

```

```

    cur = cur->next;
}
Prev->next = NULL;
free (cur);
return first;
}

Void display (Nodefirst)
{
    Node temp;
    if (first == NULL)
    {
        printf ("List is empty \n");
    }
    for (temp = first; temp != NULL; temp = temp
         → next)
    {
        printf ("%d \n", temp->item);
    }
}

Void Stack ()
{
    int choice, data;
    Node head = NULL;
    printf ("STACK IS IMPLEMENTED INSERT REAR
            AND DELETE REAR \n");
    do
    {
        printf ("1 : INSERT REAR \n");
        printf ("2 : DELETE REAR \n");
        printf ("3 : DISPLAY \n");
        printf ("4 : EXIT \n");
        printf ("Enter your choice \n");
        printf ("Scanf ("%d", &choice \n");
        Scanf ("%d", &choice);
        switch (choice)
    }
}

```

Date: / /

```
Case 1 ;
printf ("Enter The Data To Be Inserted\n");
scanf ("%d", &data);
head = insert_end (head, data);
break;
Case 2 ;
head = delete_end (head);
break;
Case 3 ;
display (head);
break;
}
while (choice != 4);
}

void q()
{
int choice, data;
Node head = NULL;
printf ("QUEUE IS IMPLEMENTED INSERT
FRONT AND DELETE FRONT\n");
do
{
printf ("1: INSERT FRONT\n");
printf ("2: DELETE FRONT\n");
printf ("3: DISPLAY\n");
printf ("4: EXIT\n");
printf ("Enter your choice\n");
scanf ("%d", &choice);
switch (choice)
{
Case 1 :
printf ("Enter The data To Be
Inserted\n");
scanf ("%d", &data);
head = insert_front (head, data);
break;
```

Date / /

Case 2 :

```
head = delete_front (head);  
break;
```

Case 3 :

```
display (head);  
break;
```

{

y

```
while (choice != 5);
```

y

Case 4 :

```
head = delete_end (head);  
break;
```

y

```
while (choice != 5);
```

y

```
int main ()
```

{

```
int option;
```

```
printf ("1: STACK\n");
```

```
printf ("2: QUEUE\n");
```

```
printf ("3: EXIT\n");
```

```
printf ("Enter your choice\n");
```

```
scanf ("%d", &option);
```

```
switch (option)
```

{

```
case 1:
```

```
stack ();
```

```
break;
```

```
case 2:
```

```
q(); break;
```

```
case 3:
```

```
exit (0);
```

```
break;
```

y}

OUTPUT:

STACK-OUTPUT

```
C:\Users\Shrehan Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\stack_and_queue.exe
1:STACK
2:QUEUE
3:EXIT
Enter Your Choice
1
STACK IS IMPLEMENTED INSERT REAR AND DELETE REAR
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
5
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
10
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
15
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
20
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
3
5
10
15
20
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
2
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
3
5
10
15
```

QUEUE-OUTPUT:

```
● "C:\Users\Shreehan Kulkarni\OneDrive\Desktop\LAB SUBJECTS\DS LAB\stack_and_queue.exe"
1:STACK
2:QUEUE
3:EXIT
Enter Your Choice
1
STACK IS IMPLEMENTED INSERT REAR AND DELETE REAR
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
5
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
10
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
15
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
1
Enter The Data To Be Inserted
20
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
2
1
10
15
20
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
2
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
Enter Your Choice
3
1
5
10
15
1:INSERT REAR
2:DELETE REAR
3:DISPLAY
4:EXIT
```

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. c) Display the contents of the list

Name: Sharachi Kulkaari

OSN: IBM17IS153

```
#include<stdio.h>
#include<math.h>
#include<cs50.h>
struct node
{
    int info;
    struct node *llink;
    struct node *rlink;
};

typedef struct node *NODE;
NODE getnode()
{
    NODE x;
    x = (NODE) malloc( sizeof( struct node));
    if (x==NULL)
        printf("memoryfull\n");
    exit(0);
}
return x;
}

void freenode(NODE x)
{
    free(x);
}

NODE insert_front(int item, NODE head)
```

```
NODE x;  
NODE temp, cur;  
temp = getnode();  
temp->info = item;  
cur = head->rlink;  
head->rlink = temp;  
temp->llink = head;  
temp->rlink = cur;  
cur->llink = temp;  
return head
```

}

```
NODE insert_end (int item, NODE head)
```

```
{  
NODE temp, cur;  
temp = getnode();  
temp->info = item;  
cur = head->rlink;  
head->rlink = temp;  
temp->llink = head;  
temp->rlink = cur;  
cur->llink = temp;  
return head
```

}

```
NODE delete_front (NODE head)
```

```
{  
NODE cur, next;  
if (head->rlink == head)
```

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

}

```

    cur = head->rlink;
    next = cur->rlink;
    head->rlink = next;
    next->llink = head;
    printf ("The node deleted is %d\n", cur->info);
    free(cur);
    return head;
}

NODE delete_rail(NODE head)
{
    NODE cur, prev;
    if (head->rlink == head)
        printf ("Empty\n");
    else
        head = head->rlink;
    cur = head->llink;
    prev = cur->llink;
    head->llink = prev;
    prev->rlink = head;
    printf ("Node deleted is %d\n", cur->info);
    free(cur);
    return head;
}

void display(NODE head)
{
    NODE limp;
    if (head->rlink == head)
        printf ("Empty\n");
    else
        limp = head;
}

```

```

    {
        printf ("Elements (%d)\n", n);
        temp = head->link;
        while (temp != head)
    {
        printf ("%d\n", temp->info);
        temp = temp->link;
    }
    printf ("%d\n", n);
}

```

NODE insert_leftpos(int item, NODE head)

 {
 NODE temp cur, prev;
 if (head->link == head)

 printf ("empty\n");
 return head;

 }
 cur = head->link;
 while (cur != head)

 {
 if (item == cur->info)

 break;

 }
 cur = cur->link;

 }
 if (cur == head)

 printf ("Key not found\n");
 return head;

}

`prev = cur -> clink;`
`printf("Enter toward left of %d = ", itm);`
`temp = getnode();`
`scanf("%d", &temp->info);`
`prev -> clink = temp;`
`temp -> clink = prev;`
`cur -> clink = temp;`
`temp -> clink = cur;`
`return head`

`}`
`NODE insert_right_pos(item, NODEhead)`

$$15 \times 16 + 15 \times 16^0$$

$$240 + 15$$

`{`
`NODE temp, cur, prev`
`if (head -> clink == head)`
`printf("empty\n");`
`return head`
`}`
`cur = head -> clink;`
`while (cur != head)`
`{`
`if (item == cur->info)`
`break;`
`cur = cur->clink;`
`}`
`if (cur == head)`
`printf("key not found\n");`
`return head`

`}`

prev = cur → llink
 printf("Enter toward right of 'l.d' = ", item)
 temp = getnode();
 scanf("%d", &temp->info);
 temp → rlink = cur → rlink
 temp → llink = cur
 cur → rlink = temp
 return head

void search(NODE head)

```

NODE ptr
int item, i=0, flag;
ptr = head;
if (ptr == NULL)
  printf("Empty list");
else
  printf("Enter item to search in ");
  scanf("%d", &item);
  while (ptr != NULL)
    if (ptr->info == item)
      printf("Item found at %d\n", i);
      flag = 1;
      break;
    else
      i++;
  }
  if (flag == 0)
    printf("Item not found");
  }
}
  
```

}

flag = 1

{

i++;

ptr = ptr -> rlink

}

if (flag == 1)

printf("Item not found\n");

break;

}

```
NODE delete_all(keyint item, NODE head)
```

```
    NODE prev, cur, next;
```

```
    int count;
```

```
    if (head == rlink == head)
```

```
        printf("Empty\n");
```

```
        return head;
```

```
}
```

```
count = 0;
```

```
cur = head == rlink;
```

```
while (cur != head)
```

```
{
```

```
    if (item == cur->info)
```

```
        cur = cur->rlink
```

```
    else
```

```
{
```

```
    found++
```

```
    prev = cur->link
```

```
    next = cur->rlink
```

```
    prev->rlink = count
```

```
    next->link = prev
```

```
    free(cur);
```

```
    cur = next;
```

```
}
```

```
if (found == 0)
```

```
    printf("Not found\n");
```

```
else
```

```
    printf("Key found at %d position (%d)",  
           count);
```

```
return head;
```

```
}
```

int main()

```

    {
        NODE head, last
        int item, choice
        head = getnode()
        head->link = head
        head->item = head
    }

```

```

    printf("1: Insert front\n");
    printf("2: Insert rear\n");
    printf("3: Delete front\n");
    printf("4: Delete rear\n");
    printf("5: display\n");
    printf("6: insert left\n");
    printf("7: RemoveDuplicates\n");
    printf("8: Search\n");
    printf("9: insert right\n");
    printf("10: exit\n");
    printf("Enter your choice\n");
    Scanf("%d", &choice)
    switch(choice)
    {

```

```

        case 1: printf("Enter item\n");
        Scanf("%d", &item);
        last = insert_front(item, head);
        break;
    }

```

```

        case 2: printf("Enter item\n");
        Scanf("%d", &item);
        last = insert_end(item, head);
        break;
    }
}

```

case 3: fast = delete_front (head);
break;

case 4: fast = delete_rear (head);
break;

case 5: display (head)
break;

case 6: printf ("Enter the item no");
scanf ("%d", &item);
head = insert_left_pos (item, head);
break;

case 7: printf ("Enter the item no");
scanf ("%d", &item);
head = delete_all_no (item, head);
break;

case 8: search (head);
break;

case 9: printf ("Enter item no");
scanf ("%d", &item);
head = insert_right_pos (item, head);
break;

default: cout (0);

}

OUTPUT

```
C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doublylinkedlist.exe

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
25

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
50

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
75

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
100

1:insert front
2:insert rear
```

```
C:\Users\Shreehan Kulkarni\OneDrive\Desktop\LAB SUBJECTS LAB\doublylinkedlist.exe
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
1
enter the item at front end
125

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125
100
75
50
25

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
6
enter the key item
100
enter towards left of 100=5

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125
```

```
C:\Users\Shreehan Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doubly\linkedlist.exe

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
9
enter the key item
100
enter towards right of 100-10

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
5
contents of dq
125
5
100
10
75
50
25

1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
8

Enter item which you want to search?
100

item found at location 3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
```

```
C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\doubly\linkedlist.exe
Enter item which you want to search?
100
item found at location 3
1:insert front
2:insert rear
3:delete front
4:delete rear
5:display
6:Insert Left
7: remove duplicates
8: Search
9:Insert Right
10:Exit
enter the choice
```

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
- c) To display the elements in the tree.

Name: Shreehari
USN: IBM19CS153

Binary Search Tree

```
#include < stdlib.h >
#include < math.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;
}
```

NODE insert (NODE root, int info)
{

NODE new_node = getnode(info)
if [new]
if

if [root == NULL]

NODE new_node = getnode(info)

else if (root → data <= info)

root → right = insert (root → right,
→ right, info)

else

root → left = insert (root → left, info)

}

```
Void preorder(NODE root)
```

```
{ if (root == NULL) {  
    printf ("%d", root->data); }
```

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

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

```
}
```

```
{
```

```
Void postorder(NODE root)
```

```
{
```

```
if (root == NULL)  
    return;
```

```
postorder (root->left);
```

```
postorder (root->right);
```

```
}
```

```

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

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

NODE delete_newnode (NODE root, data)
{
    if (root == NULL)
        return root;
    else if (info < root-> data)
        root-> left = delete_newnode
            (root-> left, data);

    else if (info > root-> data)
        root-> right = delete_newnode
            (root-> right,
             data);
}

```

if [root → right == NULL & &

NODE temp = root;

free(temp);

root == NULL;

return NULL;

else

else if (root → right == NULL)

NODE temp = root;

free(temp);

root = root → right

return root;

}

else if (left → right == NULL)

NODE temp = root;

free(temp);

root = root → right;

return root

}

$\text{root} \rightarrow \text{left} = \text{null}$)

else
{

NODE temp = findmin($\text{root} \rightarrow \text{right}$)

$\text{root} \rightarrow \text{data} = \text{temp} \rightarrow \text{data}$,

$\text{root} \rightarrow \text{right} = \text{delete node}$

($\text{root} \rightarrow \text{right}$,
 $\text{temp} \rightarrow \text{d}$)

}

return root

{

```
void display(NODE root, int i)
{
    if (root == NULL)
        return;

    display(root->right, i);
    for (int j = 1; j <= i; j++)
        printf(" ");
    printf("%d\n", root->data);
    display(root->left, i);
    display(root->left, i + 1);
}
```

```
int height(NODE root)
```

```
{
    if (root == NULL)
        return -1;
    int left_height = height(root->left);
    int right_height = height(root->right);
    int val = max(left_height, right_height);
    return val + 1;
}
```

get

int leafCount(NODE root)

}

if (root == NULL)

return 0;

if (root->right == NULL &&

root->left

== NULL)

return 1

else

return (leafCount(root->right)
+ leafCount(root->left))

}

```
int main ()
{
    printf("1: insert\n");
    do {
        NODE root = NULL;
        int option, data;
        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("Enter your option");
        scanf("%d", &option);
        switch(option)
    }
```

case 1:

```
    printf("Enter data\n");
    scanf("%d", &data);
    root = insert(root, data);
    break;
```

case 2:

```
    printf("enter data\n");
    scanf("%d", &data);
```

o root = delete_node (root, data);
break;

case 3 :
preorder (root);
break;

case 4 :
postorder (root);
break;

case 5 :
inorder (root);
break;

case 6 :
display (root);
break;

} while (option != 7);

{

{

OUT

OUTPUT:

```
[1] "C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECTS\LAB\binarysearchtree.exe"
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
15
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
10
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
25
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
1
Enter The Data To Be Inserted
35
1:Insert
2:Delete
3:Preorder
```

```
C:\Users\Shreehari Kulkarni\OneDrive\Desktop\LAB SUBJECT\DS LAB\binarysearchtree.exe"
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
6
    35
    25
    15
    10
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
3
15    10    25    35
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
4
10    35    25    15
1:Insert
2:Delete
3:Preorder
4:PostOrder
5:Inorder
6:Display
7:Height Of Tree
8:Find Maximum
9:Search
10:In Order Successor
11:Exit
Enter Your Choice
5
10    15    25    35
1:Insert
2:Delete
3:Preorder
4:PostOrder
```

11: Write a program

- To construct a binary tree.
- To traverse the tree using all the methods i.e., in-order, preorder and post order
- To display the elements in the tree.

Shrechardh Kulkarni
USN: 1BM19CS153

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

struct node
{
    int info
    struct node *llink;
    struct node *rlink;
};

typedef struct node *NODE
NODE getnode()
{
    NODE x;
    x = (NODE) malloc (sizeof (struct node));
    if (x == NULL)
        printf ("Memory not available");
    return (x);
}
```

return x;

NODE insert(int item, NODE root)

5

NODE temp, cur, prev;
 char direction[10];
 int i;
 temp = getnode();
 temp->info = item;
 temp->llink = NULL;
 temp->rlink = NULL;
 if (root == NULL)
 return temp

7:- printf("Give direction to insert in")
 scanf("%s", direction);
 prev = NULL;
 cur = root;
 for (i=0; i< strlen(direction) &&
 cur != NULL; i++)

5

prev = cur;
 if (direction[i] == 'l'):
 cur = cur->llink;
 else

$\text{cur} = \text{cur} \rightarrow \text{zlink}$

{ if ($\text{cur}' = \text{NULL}$; || $i! = \text{stolen}(\text{direction})$)

 printf ("insertion not possible");
 free (temp);
 return (root);

{ if ($\text{cur} = \text{NULL}$)

 if ($\text{direction}(i-1) == 'l'$)

 prev $\rightarrow \text{llink} = \text{temp}$

 else

 prev $\rightarrow \text{rlink} = \text{temp}$

 return (root);

}

```
void preorder(NODE root)
```

```
{
```

```
if (root != null)
```

```
{
```

```
printf("Item is %d\n", root->info)
```

```
preorder (root->elink);
```

```
preorder (root->elink);
```

```
}
```

```
}
```

```
void inorder (NODE root)
```

```
:
```

```
if (root != null)
```

```
{
```

```
inorder (root->elink);
```

```
printf ("Item is %d\n", root->info);
```

```
inorder (root->elink);
```

```
}
```

```
}
```

```
Void postorder(NODE root)
```

```
{ if (root != NULL)
```

```
    postorder (root->llink);
```

```
    postorder (root->rlink);
```

```
    printf ("% Item : %d\n", root
```

```
                  ->info)
```

```
}
```

```
}
```

```
Void display(NODE root, int i)
```

```
int j;
```

```
{ if (root != NULL)
```

```
    display (root->rlink, i+1);
```

```
    for (j = i; j < i; j++)
```

```
        printf ("%d ",
```

```
                root->info);
```

```
    display (root->llink, i+1);
```

```
}
```

Void main()

{

 NODE root = NULL

 int choice, i, item

 for(i; i)

 printf("1: insert\n");

 printf("2: preorder\n");

 printf("3: inOrder\n");

 printf("4: postorder\n");

 printf("5: Display\n");

 scanf("%d", &choice);

 switch(choice)

{

 case 1: printf("Enter item\n");

 scanf("%d", &item);

 root = insert(item, root);

 break;

 case 2: if (root == NULL)

 printf("tree empty\n");

{

```
else
{
    printf("Given tree is ");
    display(root, 1);
    printf(" Preorder traversal");
    preorder(root);
}
```

break;

case 3: if (root == NULL)

```

{
    printf("Tree empty");
}
```

else

```
{
```

```
    printf("Given tree is ");
    display(root, 1);
    printf(" in order traversal");
    inorder(root);
}
```

break;

case 4: if (root == NULL)

{ } b

printf ("Tree empty\n");

{
else
}

printf ("Element is ");

display (root, 1);

printf (" postorder ");

postorder (root);

{

break;

case 5: display (root);

break;

default: exit (0);

{

{

{

OUTPUT:

```
C:\Users\Shreesh Kulkarni\OneDrive\Desktop\LAB SUBJECTS\DS LAB\Binarytree.exe"
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
25
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
5
give direction to insert
1
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
1
enter the item
48
give direction to insert
r
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
5
    48
    25
    5
1.insert
2.preorder
3.inorder
4.postorder
5.display
enter the choice
```