VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

DATA STRUCTURE LAB RECORD

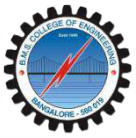
Submitted by

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 (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

LAB 1 program

NAP to simulate the working of stack using an array

- a push
- pop
- display

Program should print appropriate message for stack overflow and underflow.

Page No.

Date / /

```
#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 ;
    if (top==-1)
    {
      printf(" Stack is empty\n");
      return;
    }
}
```

Saathi

Date __/__/____

```
for(i = top ; i >= 0; i--)
{
    printf("%d\n", s[i]);
}
}

void main()
{
    int item_deted
    int choice;
    for(;;)
    {
      printf("\n1: Push\n2: Pop\n3: display
                \n4: exit\n");

      printf("Enter the choice\n");
      switch(choice)
      {
        case 1:
        printf("Enter the item to be inserted
                \n");

        scanf("%d", &item);
        push()
        break;
```

Saathi

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:
    exit(0)
}
}
```

Page No.

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 insated
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

Date __/__/____

Saathi

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

4.

Saathi

Date __/__/____

Week2: Write a program to convert infix to postfix using stacks.

```
#include<stdio.h>
#include<string.h>
int F(char symbol)
{
    switch(symbol)
    {
        case '+':
        case '-':
            return 2;

        case '*':
        case '/':
            return 4;

        case '^':
        case '$':
            return 5;

        case '(':
            return 0;

        case '#'
            return -1;

        default:
            return 8
    }

}
```

Page No.

Date __/__/____

Saathi

```
int G(char symbol)
{
    switch(symbol)
    {

      case '+':
      case '-':
        return 1;

      case '*'
      case '/'
        return 3;

      case '^':
      case '$':
        return 6;

      case 'c':
        return 9;

      case '('
        return 0

      default
        return 7

    }

}
```

Saathi

Date / /

```
void infix_postfix(char infix[], char postfix[])
{
    int top, i, j;
    char s[30], 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 __/__/____

Saathi

(1)
```
printf("Enter the valid expression\n");
scanf("%s", infix);
infix_postfix(infix, postfix);
printf("postfix is \n");
printf("%s\n", postfix);
}
```

OUTPUT

Enter the valid Expression

(A + (B-C) * D)

Post fix Expression is

ABC - D* + .

WEEK 5

Date

Saathi

I mplementing 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 ("\n1: Insert\n");
      printf ("\n2: Delete\n");
      printf ("\n3: Display\n");
      printf ("Enter your option\n");
      scanf ("%d", &option);
      switch (option)
      {
          case 1:
              insert()
              break;

          case 2:
              Vale = delet();
              if (Val != -1)
              printf ("Deleted Number is %d\n",
                                  Val);
              break;

          case3: display()
              break
  } }
```

Page No.

Saathi

Date ___/___/___

```
{ while (option != 5)
  return 0

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 ("\n 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)
     print ("\n Queue is empty");
  else
  {
     if (front < rear)
```

Saathi

Date ___/___/_____

```
{
    for( int i = front; i<max; i++)
        printf("%d", queue[i]);
    for(i=0;
}
else
{
    for(int i = front; i<Max; i++)
        printf("%d", queue[i]);
    for(int i=0; i<= rear; i++)
        printf("%d\n", queue[i]);
}
}
}
```

Date ___/___/_____

Saathi

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

10

Saathi

Date __/__/__

Implementing Linear Queue

```
#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("Item deleted
                is: %d\n", val);
```

```
        break;

    case 3:
            display();
            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
    {
        rear++;
        q[rear]=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
    }
}

void 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
1
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
6

Date __/__/____

Saathi

linkdist programm

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

Struct node
{
  intdata;
 Struct node *next;
};
 typedef struct node *NODE;

 Void swap (NODE a, NODE b);
 NODE getNode()
{
   NODE x;
   x = malloc (sizeof (structnode));
    if (x = NULL)
   {
      exit(0);
   }
    else
   } return x;
     }
}
   NODE insert_front (int item, NODE
                                 start)
{     NODE temp;
      temp = getNode();
      temp -> data = item;
      temp -> next = start;
```

Page No.

Date __/__/__

Saathi

```
start = temp;
return start;
}
NODE insert_end (int item, NODE start)
{
NODE insert_end (int item, NODE start)
{
NODE temp, cur;
temp = getNode();
temp -> data = item;
temp -> next = NULL;
cur = start;
while (cur -> next != NULL)
{
cur = cur -> next;
}
cur -> next = temp;
return start;
}
NODE insert_pos (int pos, int item,
                 NODE start)
{ NODE temp, prev, cur;
temp = getNode();
temp -> data = item;
temp -> next = NULL;
if (pos == 1 && start == NULL)
{
return temp;
}
if (start == NULL)
{
printf ("Invalid position\n");
return start;
}
if (pos == 1)
```

Saathi

Date __/__/____

```
}
    temp -> next = start;
    return temp;
}
    int count = 1;
    cur = start;
    prev = NULL;
    while (cur != NULL && count != pos)
    {
        prev = cur;
        cur = cur -> next;
        count++;
    }
    if (count == pos)
    {
        prev -> next = temp;
        temp -> next = cur;
        return start;
    }
    Printf ("Invalid position \n");
    return start;
}

    NODE temp;
    if (start == NULL)
    {
        printf ("List Is empty And Cannot Be
                 Displayed \n");
        exit (0);
    }
        temp = start;
        Printf("items are \n");
        printf (" \n");
    for (temp = start ; temp != NULL;
```

Saathi

Date __/__/____

```
temp = temp -> next)
{
  Printf ("%d\n", temp -> data);
}
Printf ("\n");
return Start;
}
Void bubbleSort (NODE Start)
{
 int swapped, i;
 NODE cur;
 NODE Prev = NULL)
{
  Printf ("Empty LinKed List\n");
  Return;
}
  do
{
  Swapped = 0;
  cur = Start;
 while (cur -> next != Prev)
{
  if (cur -> data > cur -> next -> data)
  }
  Swap (cur, cur -> next);
  Swapped = 1;
  }
  Cur = cur -> next;
}
  Prev = cur;
}
 while (swapped);
}
 Void swap (NODE a, NODE b)
```

Saathi
Date __/__/____

```
{
    int temp = a -> data;
    a -> data = b -> data;
    b -> data = temp;
}
Void reverseList (NODE Start)
{
    NODE Prev, Cur;
    if (Start != NULL)
    {
     Prev = Start;
     Start = Start -> next;
     cur = Start;
     Prev ==> next = NULL; // Make first Node as
                              last node
    while (Start != NULL)
    {
     Start = Start -> next;
     Cur -> next = Prev;
     Prev = Cur;
     Cur = Start;
    }
    Start = Prev; // make last node as head
    Print ("SUCCESSFULLY REVERSED) LISTIN");
}
    NODE temp;
    temp = Start;
    printf ("Reversed List is \n");
    while (temp != NULL)
    {
     Printf ("%d\n", temp -> data);
     temp = temp -> next;
    }
}
```

Saathi

Date __/__/____

```
int main()
{
 int option, item, pos;
 NODE start = NULL;
 do
 {
 Printf("1: Insert Front\n");
 Printf("2: Insert End\n");
 Printf("3: Insert At particular position\n");
 Printf("4: Delete First\n");
 printf("5: Delete End\n");
 Printf("6: Delete At particular position\n");
 Printf("7: Display\n");
 Printf("8: Sort\n");
 Printf("9: Reverse\n");
 Printf("10: exit\n");
 Printf("Enter your choice\n");
 Scanf (~~option~~) ("%d", &option);
 Switch (option)
 {
  case 1:
   printf("Enter the item to be inserted\n");
   Scanf("%d", &item);
   start = insert_front(item, start);
   break;
  case 2:
        printf("Enter the item to be
                inserted\n");
       Scanf("%d", &item);
       Start = insert_end(item, start);
        break;
    case 3:
    Printf("Enter the item to be inserted
```

Saathi

Date / /

```
scanf ("%d", &pos);
start = insert_pos (pos, item, start);
break;

case 4;
    start = delete_front (start);
    break;

case 5:
    start_delete_end (start);
    break;

case 6:
    printf ("Enter the position where element
             has to be inserted \n");
    scanf ("%d", &pos);
    start = delete_pos (pos, start);
    break;

case 7:
    start = display (start);
    break;

case 8:
    bubblesoart (start);
printf (" Items In Sorted order are\n
start = display (start);
break;

case 9;
    reverselist (start);
    break;
}
} while (option != 10);
}
```

Saathi

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;
}
void sort(Node first)
{
int t;
Node temp;
if(first==NULL)
{
printf("List IS Empty\n");
return;
}
for(Node i=first; i!=NULL; i=i->next)
{
for(Node j=i->next; j!=NULL
                j=j->next)
{
if((i->item) > (j->item))
```

Saathi

Date / /

```
    t = i -> item;
    i -> item = j -> item
    j -> item = t;
  }
}

Node concat (Node first, Node second)
{
  Node cur;
  if (first == NULL)
  {
    first = second;
    return first;
  }
  if (second == NULL)
  {
    return first;
  }
  cur = first;
  while (cur -> next != NULL)
  {
    cur = cur -> next;
  }
  cur -> next = second;
  printf("Final Concatinated List is\n");
  return first;
}
```

Date ___/___/___

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

struct node
{
    int item;
    struct 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;
}
```

Date ___/___/___

Saathi

```
Node delete_end(Node first)
{
    Node prev, cur;
    if (first == NULL)
    {
        printf("underflow\n");
        return first;
    }
    cur = first;
    while (cur -> next != NULL)
    {
        prev = cur;
        cur = cur -> next;
    }
    prev -> next = NULL;
    free(cur);
    return first;
}
Void search(Node first, int data)
{
    int pos = 0;
    Node temp;
    int i;
    if (first == NULL)
    {
        printf("underflow\n");
        return;
    }
}
```

```
for (temp = first; i=0; temp != NULL;
                        temp = temp->next;
                                    i++)
{
    if (temp -> item == data)
    {
        pos = i + 1;
        printf ("Search successful \n");
        printf ("Element found at %d \n",
                                    pos);
        break;
    }
    else
    {
        pos = 0;
    }
}
if (pos == 0)
{
    printf ("Search unsuccessful \n");
}
}
```

Date ___/___/___

Saathi

```
void sort (Node first)
{
    int t;
    Node temp;
    if (first == NULL)
    {
        printf ("list empty\n");
        return;
    }
    for (Node i = first; i != NULL; i = i->next)
        for (Node j = i->next; j != NULL; j = j->next)
            if ((i->item) > (j->item))
            {
                t = i->item;
                i->item = j->item;
                j->item = t;
            }
    printf ("list in sorted order is \n");
}
```

Saathi

Date ___/___/_____

```
void display(Node first)
{
    int count=0;
    Node temp;
    if (first == NULL)
    {
        printf("underflow\n");
        return;
    }
    for (temp= first; temp != NULL; temp= temp->next)
    {
        count++;
        printf("%d\n", temp->item);
    }
    printf("Number of Nodes %d\n", count);
}

int main()
{
    Node first = NULL;
    int val, n;
    do
    {
        printf("1: Insert at front\n");
        printf("2: Delete Rear\n");
        printf("3: Sort\n");
```

Date __/__/__

Saathi

```
printf("4: Search\n");
printf("5: Display\n");
printf("6: exit\n");
printf("Enter your choice\n");
scanf("%d", &choice);
switch(choice)
{
  case 1:
      printf("Enter the value to be inserted\n");
      scanf("%d", &val);
      first = insert_front(first, val);
      break;

  case 2:
      first = delete_end(first);
      break;

  case 3:
      sort(first);
      break;

  case 4:
      printf("Enter the element to be searched\n");
      scanf("%d", &n);
      search(first, n);
      break;

  case 5:
      display(first);
      break;
}while(option != 6);
```

Page No.

Name: Shreehari Kulkarni
USN: 1BM19CS153

```
#include<stdio.h>
#include<math.h>
#include<stdio.h>
struct node
{
    int info
    struct node *rlink;
    struct node *llink;
};
typedef struct node * NODE
NODE getnode()
{
    NODE x;
    x = (NODE) malloc( sizeof( struct node));
    if (x==NULL)
    {
        printf ("memory full\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_rear(NODE head)
{
    NODE cur, prev;
    if(head->rlink == head)
    {
        printf("Empty\n");
        return head;
    }
    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 temp;
    if(head->rlink == head)
    {
        printf("Empty\n");
        return;
    }
```

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

NODE insert_leftpos(int item, NODE head)
{
    NODE temp, cur, prev;
    if (head -> rlink == head)
    {
        printf ("Empty \n");
        return head;
    }
    cur = head -> rlink;
    while (cur != head)
    {
        if (item == cur -> info)
        {
            break;
        }
        cur = cur -> rlink;
    }
    if (cur == head)
    {
        printf ("Key not found \n");
        return head;
    }
```

prev = cur -> llink;
printf("Enter toward left of %d = ", item);
temp = getnode();
scanf("%d", &temp->info);
prev -> rlink = temp;
temp -> llink = prev;
cur -> llink = temp;
temp -> rlink = cur;
return head
}

NODE insert_right_pos(int item, NODE head

15 x 16 + 15 x 16^0

240 + 15.

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

}

```
prev = cur -> llink
printf("Enter toward right of %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\n");
    }
    else
    {
        printf("Enter item to search\n");
        Scanf("%d", & item);
        while (ptr != NULL)
        {
            if (ptr -> info == item)
            {
                printf("Item found at %d\n", i);
                flag = 0;
                break
            }
            else
```

```
        {
            flag = 1
        {
        i++;
        ptr = ptr -> rlink
    }
if (flag == 1)
{
    printf("\nItem not found\n");
    break;
}
```

```
NODE delete_all(int 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
        {
            count++;
            prev = cur -> llink;
            next = cur -> rlink;
            prev -> rlink = next;
            next -> llink = prev;
            free(cur);
            cur = next;
        }
    }
    if(count == 0)
        printf("Not found\n");
    else
        printf("Key found at %d position\n", count);
    return head;
}
```

```
int main()
{
    NODE head, last
    int item, choice
    head = getnode()
    head->rlink = head
    head->llink = head
    for(;;)
    {
        printf("1: Insert front\n");
        printf("2: Insert Rear\n");
        printf("3: Delete front\n");
        printf("4: DeleteRear\n");
        printf("5: display\n");
        printf("6: insert_left\n");
        printf("7: RemoveDuplicates\n");
        printf("8: search\n");
        printf("9: insertRight\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 = dinsert_front(item, head);
                    break

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

```
case 3: last = delete_front (head);
        break;

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

case 5: display(head)
        break;

case 6: printf("Enter the item\n");
        scanf("%d", &item);
        head = insert_leftpos(item, head);
        break;

case 7: printf("Enter the item\n");
        scanf("%d", &item);
        head = delete_all_key(item, head);
        break;

case 8: search(head);
        break;

case 9: printf("Enter item\n");
        scanf("%d", &item);
        head = insert_rightpos(item, head);
        break;

    default: exit(0);

}
}
```

Name: Shreeharsh
USN: 1BM19CS153

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 (root == NULL)
        NODE new_node = getnode(info)

else if (root -> data <= info)
        root -> right = insert(root
                               -> right, info

else
    root -> left = insert(root -> left, inf
);
}
```

```
Void preorder(NODE root)
{ if (root != NULL) {
    printf("%d\t", root->data);
    preorder(root->left);
    preorder(root->right);
  }
}

Void postorder(NODE root)
{
    if (root == NULL)
        return

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

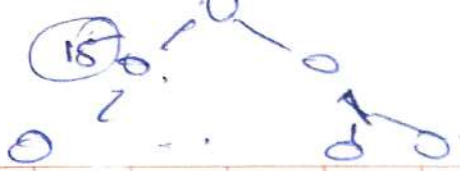
```
void inorder (NODE root)
{
    if (root == NULL)
        return

    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)

else
```

```
if (root -> right == NULL &&
{
    NODE temp = root;
    free(temp);
    root == NULL;
    return true;
}
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
}
```



Page No.
Date / /

root -> left == NULL)

else
{

NODE temp = findmin(root -> right)

root -> data = temp -> data;
root -> right = delete
(root -> right-
, temp -> d

}
}return root

{

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

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

Page No.
Date / /

getleafCount(NODE root)

```
int leafCount(NODE root)
{
    if (root == NULL)
        return 0;
    if (root->right == NULL && root->left == NULL)
        return 1
    else
        return (getCount(root->right) + getLeafCount(root->left));
}
```

```
int main()
{
    printf("1:insert
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);
```

```
            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);
    }
}
```

Shreeharsi 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");
       exit(0);
    }
```

```
return x;

NODE insert(int item, NODE root)
{
    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
    printf("Give direction to insert\n");
    scanf("%s", direction);
    prev = NULL;
    cur = root;
    for(i=0; i< strlen(direction) &&
                cur != NULL;
                i++)
    {
        prev = cur;
        if(direction[i] == 'l')
            cur = cur->llink;
        else
```

```
cur = cur -> rlink
}
if ( cur != NULL || i != strlen(direction)
    printf("insertion not possible");
    free(temp);
    return(root);
{
if (cur == NULL)
    if (direction[i-1] == 'l')
        prev -> llink = temp
    else
        prev -> rlink = temp
;
return(root);
}
```

```
void preorder(NODE root)
{
  if (root != NULL)
  {
    printf("Item is %d\n", root->info);
    preorder (root -> llink);
    preorder (root -> rlink);
  }
}

void inorder (NODE root)
{
  if (root != NULL)
  {
    inorder (root -> llink);
    printf ("Item is %d\n", root->info);
    inorder (root -> rlink);
  }
}
```

```
void postorder(NODE root)
{
    if(root != NULL)
    {
        postorder(root->llink);
        postorder(root->rlink);
        printf("Item is %d\n", root
                                ->info);
    }
}

void display(NODE root, int i)
{
    int j;
    if(root != NULL)
    {
        display(root->rlink, i+1);
        for(j=1; j<=i; j++)
            printf("  ");
        printf("%d\n", root->info);
        display(root->llink, i+1);
    }
}
```

```
void main()
{
    NODE root = NULL
    int choice, i, item
    for(;;)
    {
        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)
{
    case1:  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\n");
    preorder(root);
}

break;

case 3: if(root==NULL)
{
    printf("Tree empty\n");
}
else
{
    printf("Given Tree is");
    display(root, 1);
    printf("In order traversal\n");
    inorder(root);
}
break;
```

```
case 4: if (root == NULL)
        {
            printf("\nTree empty\n");
        }
        else
        {
            printf("\nEnter tree is");
            display(root, 1);
            printf("\n postorder traversal\n");
            postorder(root);
        }
        break;

case 5: display(root, 1);
        break;

default: exit(0);
    }
}
```