DEPARTMENT OF COMPUTER SCIENCE & BUSINESS SYSTEMS

Vision

To create a competent learning ecosystem that deepens and advances the understanding of technology and management to develop innovative, principled, and insightful leaders who can change the world.

Mission

- To imbibe analytical and critical thinking skills for problem solving.
- To include best teaching-learning practices.
- To create an ecosystem for innovation, multidisciplinary research, and skill enhancement in disruptive technologies.
- To collaborate with industry partners to make students industry-ready.
- To develop professional leaders with good values and ethics.

Program Specific Outcomes (PSOs):

1. To create, select, and apply appropriate techniques, resources, modern engineering and business tools including prediction and data analytics to complex engineering activities and business solutions.
2. To evolve Computer Science domain specific methodologies for effective decision making in several domains like business processes and other domains.
3. To manage complex IT projects with consideration of the human, financial, ethical and environmental factors and an understanding of risk management processes, and operational and policy implications.

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to the starting time), those who come after 5 minutes will not be allowed into the lab.

2. Plan your task properly much before to the commencement, come prepared to the lab with the synopsis / program / experiment details.

3. Student should enter into the laboratory with:

a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm, Procedure, Program, Expected Output, etc.,) filled in for the lab session.

b. Laboratory Record updated up to the last session experiments and other utensils (if any) needed in the lab.

c. Proper Dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.

5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.

6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.

7. Computer labs are established with sophisticated and high end branded systems, which should be utilized properly.

8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.

9. Students must take the permission of the faculty in case of any urgency to go out ; if anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.

10. Students should LOG OFF/ SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Lab Program 1:

Design, Develop and Implement a menu driven Program in C for the following Array operations

a. Creating an Array of N Integer Elements

b. Display of Array Elements with Suitable Headings

c. Inserting an Element (ELEM) at a given valid Position (POS)

d. Deleting an Element at a given valid Position(POS)

e. Exit.

Support the program with functions for each of the above operations.

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 5
int a[MAX], pos, elem;
int n = 0;
void create();
void display();
void insert();
void delete();
void main()
{
        int choice;
        while(1)
                {
                        printf("\n\n~~~~MENU~~~~");
                        printf("\n=>1. Create an array of N integers");
                        printf("\n=>2. Display of array elements");
                        printf("\n=>3. Insert ELEM at a given POS");
                         printf("\n=>4. Delete an element at a given POS");
                         printf("\n=>5. Exit");
                         printf("\nEnter your choice: ");
```

```
                scanf("%d", &choice);
                switch(choice)
                {
                        case 1:         create();
                                        break;
                        case 2:         display();
                                        break;
                        case 3:         insert();
                                        break;
                        case 4:         delete();
                                        break;
                        case 5:         exit(1);
                                        break;
                        default:        printf("\nPlease enter a valid choice:");
                }
        }
}
void create()
{
        int i;
        printf("\nEnter the number of elements: ");
        scanf("%d", &n);
        printf("\nEnter the elements: ");
        for(i=0; i<n; i++)
    {
                scanf("%d", &a[i]);
        }
}

void display()
{
```

```
                int i;
                if(n == 0)
                {
                        printf("\nNo elements to display");
                        return;
                }
                printf("\nArray elements are: ");
                for(i=0; i<n;i++)
                        printf("%d\t ", a[i]);
}
void insert()
{
                int i;

                if(n == MAX)
                {
                        printf("\nArray is full. Insertion is not possible");
                        return;
                }

                do
                {
                        printf("\nEnter a valid position where element to be inserted:       ");
                        scanf("%d", &pos);
                }while(pos > n);

                printf("\nEnter the value to be inserted:   ");
        scanf("%d", &elem);

                for(i=n-1; i>=pos ; i--)
                {
```

```
            a[i+1] = a[i];
        }
        a[pos] = elem;
        n = n+1;
        display();
}
void delete()
{
        int i;

    if(n == 0)
        {
            printf("\nArray is empty and no elements to delete");
            return;
        }

        do
        {
            printf("\nEnter a valid position from where element to be deleted:   ");
            scanf("%d", &pos);
        }while(pos>=n);

        elem = a[pos];

        printf("\nDeleted element is : %d \n", elem);
        for( i = pos; i< n-1; i++)
        {
            a[i] = a[i+1];
        }
        n = n-1;
    display();
```

}

Output:

```
~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 1

Enter the number of elements: 3
Enter the elements: 10 20 30

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 2

Array elements are: 10   20          30

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
```

```
Enter your choice: 3

Enter a valid position where element to be inserted:     5
Enter a valid position where element to be inserted:     4
Enter a valid position where element to be inserted:     3
Enter the value to be inserted:   40

Array elements are: 10   20        30     40
~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 3

Enter a valid position where element to be inserted:     4
Enter the value to be inserted:   50

Array elements are: 10   20        30     40     50

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 3

Array is full. Insertion is not possible
```

```
~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4

Enter a valid position from where element to be deleted:5
Enter a valid position from where element to be deleted:6
Enter a valid position from where element to be deleted:4
Deleted element is: 50

Array elements are: 10   20          30       40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4

Enter a valid position from where element to be deleted:2
Deleted element is: 30

Array elements are: 10   20          40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
```

```
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4

Enter a valid position from where element to be deleted:1
Deleted element is: 20

Array elements are: 10   40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4

Enter a valid position from where element to be deleted:0
Deleted element is: 10

Array elements are: 40

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4
```

```
Enter a valid position from where element to be deleted:0
Deleted element is: 40

No elements to display
~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 4

Array is empty and no elements to delete

~~~~MENU~~~~
=>1. Create an array of N integers
=>2. Display of array elements
=>3. Insert ELEM at a given POS
=>4. Delete an element at a given POS
=>5. Exit
Enter your choice: 5
```

Lab Program 2:

Design, Develop and Implement a Program in C for the following operations on Strings

a. Read a main String (STR), a Pattern String (PAT) and a Replace String (REP)

b. Perform Pattern Matching Operation: Find and Replace all occurrences of PAT in STR with REP if PAT exists in STR.

c. Report suitable messages in case PAT does not exist in STR

Support the program with functions for each of the above operations. Don't use Built-in functions.

```
#include<stdio.h>
char str[50], pat[20], rep[20], ans[50];
int     c=0, m=0, i=0, j=0, k, flag=0;
void stringmatch()
{
                 while(str[c] !='\0')
                {
                        if(str[m] == pat[i])
                        {
                                i++;
                                        m++;
                                        if(pat[i] == '\0')
                                {
                                                flag = 1;
                                                for(k=0; rep[k]!='\0'; k++, j++)
                                                {
                                                        ans[j] = rep[k];
                                                }
                                                i = 0;
                                                c = m;
                                        }
                        }
                        else
                        {
                                ans[j]= str[c];
                                j++;
                                c++;
                                m=c;
                                i=0;
```

```
                    }
            }
            ans[j]='\0';
}
void main()
{
            printf("\nEnter the main string:");
            gets(str);
            printf("\nEnter the pat string:");
            gets(pat);
            printf("\nEnter the replace string:");
            gets(rep);
        stringmatch();
            if(flag == 1)
                    printf("\nResultant string is %s", ans);
            else
                    printf("\nPattern string is not found");
}
```

```
Enter the main string:   hello aabhii howaab
Enter the pat string:     aab
Enter the replace string:                    klmno

Resultant string is: hello klmnohii howklmno
```

Method 2:

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

```
char str[50], pat[50], rep[50];
int start = 0, patfound = 0;
int lasts, lastp, lastr;

void replacepattern()
{
        int i, j;
                lastr = strlen(rep)-2;

        if(lastp != lastr)
        {
                        printf("\nInvalid length of replace string");
                        exit(0);
                }
                else
                {
                        i = start;
                        for(j=0; j<=lastr; j++)
                        {
                        str[i] = rep[j];
                                i++;
                        }
                }
                return;
}

void findpattern()
{
                int i, j, inmatch;
                lasts = (strlen(str))-2;
                lastp = (strlen(pat))-2;
                int endmatch;

                for(endmatch = lastp; endmatch<=lasts; endmatch++, start++)
                {
                        if(str[endmatch] == pat[lastp])
                        {
                        inmatch = start;
                         j=0;
                        while(j<lastp)
                        {
                                if(str[inmatch] == pat[j])
                                {
                                inmatch++;
                                    j++;
                                }
                                else
                                {
                                    break;
                                }
                        }
                        if(j == lastp)
                        {
                                patfound = 1;
                                replacepattern();
                        }
                        }
                }
                return;
}

void main()
```

```
{
                printf("\nEnter the main string(STR): ");
                fgets(str, 50, stdin);

                printf("\nEnter the pattern to be matched(PAT): ");
                fgets(pat, 50, stdin);

                printf("\nEnter the string to be replaced(REP): ");
                fgets(rep, 50, stdin);

                printf("\nThe string before pattern match is:\n %s", str);

                findpattern();

                if(patfound == 0)
                        printf("\nThe pattern is not found in the main string");
                else
                        printf("\nThe string after pattern match and replace is: \n %s ",str);
        return;
}
```

Output:

Case 1:

Enter the main string(STR): **Hello hii how are you hii**

Enter the pattern to be matched(PAT): **hii**

Enter the string to be replaced(REP): **xyz**

The string before pattern match is:
 Hello hii how are you hii

The string after pattern match and replace is:
 Hello xyz how are you xyz

<u>Case 2:</u>

Enter the main string(STR): **Hello hii how are you**

Enter the pattern to be matched(PAT): **abc**

Enter the string to be replaced(REP): **xyz**

The string before pattern match is:
 Hello hii how are you

The pattern is not found in the main string

3. Design, Develop and Implement a menu driven program in c for the following operations on STACK of integers(Array implementation of stack with maximum size MAX)
a. Push an element onto the stack.
b. Pop an element from the stack.
c. Demonstrate how stack can be used to check palindrome.

d. Demonstrate overflow and underflow situations on stack.

e. Display the status of the stack.

f. Exit

Support the program with appropriate functions for each of the above operations.

```
#include<stdio.h>
#include<conio.h>
#include<math.h>
#define max 5
int s[max],stop;
int ele,st[max],sp,ch;
void push(int ele,int s[],int *stop)
{
if(*stop>=max-1)
printf("stock overflow\n");
else
s[++*stop]=ele;
}
int pop(int s[],int *top)
{
if(*top==-1)
{
printf("stock empty\underflow\n");
return 0;
}
else
return(s[(*top)--]);
}
void palindrome(int ele,int st[])
{
int rem,rev=0,temp=ele,i=0;
```

```
while(temp!=0)
{
rem=temp%10;push(rem,st,&sp);
temp=temp/10;
}
while(sp!=-1)
rev=rev+(pop(st,&sp)*pow(10,i++));
if(ele==rev)
printf("palendrome\n");
else
printf("not a palindrome\n");
```

```
}
void display(int s[],int *stop)
{
int i;
if(*stop==-1)
printf("stalk is empty\n");
else
for(i=*stop;i>-1;i--)
printf("%d\n",s[i]);
}
void main()
{
stop= -1;
sp= -1;
while(1)
{
printf("enter tne choice\n");
printf("enter 1 to insert an element into the STACK\n");
printf("enter 2 to delete an element from the STACK\n");
printf("enter 3 to check an element is palindrome or not\n");
printf("enter 4 to check the status of the STACK\n");
printf("enter 5 to exit\n");
scanf("%d",&ch);
switch(ch)
{
case 1:printf("enter the elements to de inserted to STACK\n");
 scanf("%d",&ele);
 push(ele,s,&stop);
 break;
case 2:ele=pop(s,&stop);
 if(ele!=0)
 printf("element poped is %d\n",ele);
 break;
case 3:printf("enter the elements to chech weather it is a palindrome\n");
 scanf("%d",&ele);
 palindrome(ele,st);
 break;
case 4:printf("the status of the STACK \n");
 display(s,&stop);
 break;
case 5:exit(0);
}
}
}
```

Lab Program 4:

Design, Develop and Implement a Program in C for converting an Infix Expression to Postfix Expression. Program should support for both parenthesized and free parenthesized

expressions with the operators: +, -, *, /, %(Remainder), ^(Power) and alphanumeric operands.

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

void evaluate();
void push(char);
char pop();
int prec(char);

char infix[30], postfix[30], stack[30];
int top = -1;

void main()
{
	printf("\nEnter the valid infix expression:\t");
	scanf("%s", infix);
	evaluate();
	printf("\nThe entered infix expression is :\n %s \n", infix);
	printf("\nThe corresponding postfix expression is :\n %s \n", postfix);
}

void evaluate()
{
	int i = 0, j = 0;
	char symb, temp;

	push('#');

	for(i=0; infix[i] != '\0'; i++)
	{
 }

symb = infix[i];
switch(symb)
{
        case '(' :       push(symb);
                         break;

        case ')' :       temp = pop();
                         while(temp != '(' )
                         {
                                         postfix[j] = temp;
                                 j++;
                                         temp = pop();
                         }
                         break;
        case '+' :
        case '-' :
        case '*' :
        case '/' :
        case '%' :
        case '^' :
        case '$'  :      while( prec(stack[top]) >= prec(symb) )
                         {
                                 temp = pop();
                                         postfix[j] = temp;
                                         j++;
                         }
                         push(symb);
                         break;
        default:         postfix[j] = symb;
                         j++;
 }
```

```
        }
        while(top > 0)
        {
                temp = pop();
                postfix[j] = temp;
                j++;
        }
        postfix[j] = '\0';
}

 void push(char item)
{
        top = top+1;
        stack[top] = item;
}

 char pop()
{
        char item;
        item = stack[top];
        top = top-1;
        return item;
}
int prec(char symb)
{
        int p;
        switch(symb)
        {
                        case '#' :      p = -1;
                                break;
                case '(' :
            case ')' :          p = 0;
                                break;

            case '+' :
            case '-' :          p = 1;
                                break;

            case '*' :
            case '/' :
            case '%' :          p = 2;
                                break;

            case '^' :
            case '$' :          p = 3;
                                break;
        }
        return p;
}
```

<u>Output:</u>

```
Enter the valid infix expression:       (a+b)+c/d*e
The entered infix expression is :
 (a+b)+c/d*e
The corresponding postfix expression is :
 ab+cd/e*+
```

<u>Lab Program 5a:</u>

Design, Develop and Implement a Program in C for the following Stack Applications

a. Evaluation of Suffix expression with single digit operands and operators: +, -, *, /, %, ^

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

```
#include<math.h>

int i, top = -1;
int op1, op2, res, s[20];
char postfix[90], symb;

void push(int item)
{
        top = top+1;
                s[top] = item;
}

int pop()
{
                int item;
                item  =  s[top];
                top = top-1;
        return item;
}
void main()
{
        printf("\nEnter a valid postfix expression:\n");
        scanf("%s", postfix);
        for(i=0; postfix[i]!='\0'; i++)
        {
                symb = postfix[i];
                if(isdigit(symb))
                {
                        push(symb - '0');
                }
                else
                {
                        op2 = pop();
                        op1 = pop();
                        switch(symb)
                        {
                                        case '+':       push(op1+op2);
                                                        break;
                                        case '-':       push(op1-op2);
                                                        break;
                                        case '*':       push(op1*op2);
                                                        break;
                                        case '/':       push(op1/op2);
                                                        break;
                                        case '%':       push(op1%op2);
                                                        break;
                                        case '$':
                                        case '^':       push(pow(op1, op2));
                                                        break;
                                        default :   push(0);
                        }
                }
        }
        res = pop();
        printf("\n Result = %d", res);
}
```

<u>Output:</u>

To compile in Linux: gcc –lm 5.c

```
Enter a valid postfix expression:
623+-382/+*2$3+
Result = 52
Enter a valid postfix expression:
```

42$3*3-84/11+/+

Result = 46

<u>**Lab Program 6:**</u>

Design, Develop and Implement a menu driven Program in C for the following operations on Circular QUEUE of Characters (Array Implementation of Queue with maximum size MAX)

a. Insert an Element on to Circular QUEUE

b. Delete an Element from Circular QUEUE

c. Demonstrate Overflow and Underflow situations on Circular QUEUE

d. Display the status of Circular QUEUE

e. Exit

Support the program with appropriate functions for each of the above operations

```
#include <stdio.h>
#include<stdlib.h>
#include<stdio_ext.h>
#define MAX 3
char cq[MAX];
int front = -1, rear = -1;
void insert(char);
void delete();
void display();
void main()
{
        int ch;
        char item;
        while(1)
        {
                printf("\n\n~~Main Menu~~");
                printf("\n==> 1. Insertion and Overflow Demo");
                printf("\n==> 2. Deletion and Underflow Demo");
                printf("\n==> 3. Display");
                printf("\n==> 4. Exit");
                printf("\nEnter Your Choice: ");
                scanf("%d", &ch);
                __fpurge(stdin);
               switch(ch)
               {
                        case 1:     printf("\n\nEnter the element to be inserted: ");
                                    scanf("%c", &item);
                                    insert(item);
                                    break;
                        case 2:     delete();
                                    break;
                        case 3:     display();
                                    break;
                         case 4:    exit(0);
                         default:   printf("\n\nPlease enter a valid choice");
                }
        }
}
void insert(char item)
{
        if(front == (rear+1)%MAX)
        {
                printf("\n\n~~Circular Queue Overflow~~");
        }
        else
        {
               if(front == -1)
                        front = rear = 0;
                        else
                                rear = (rear+1)%MAX;
                        cq[rear] = item;
                }
}
void delete()
{
                char item;
                if(front == -1)
                {
                        printf("\n\n~~Circular Queue Underflow~~");
                }
                else
                {
                        item = cq[front];
                        printf("\n\nDeleted element from the queue is: %c ",item );

                        if(front == rear) //only one element
                                front = rear = -1;
                        else
                                front = (front+1)%MAX;
                }
}


void display ()
{
                int i ;
                if(front == -1)
                {
                        printf("\n\nCircular Queue Empty");
                }
                else
                {
                     printf("\nCircular Queue contents are:\n");
                     printf("Front[%d]-> ", front);
                    for(i = front; i != rear ; i = (i+1)%MAX)
                    {
                            printf(" %c", cq[i]);
                    }
                    printf(" %c", cq[i]);
                    printf(" <-[%d]Rear", rear);
                 }
}
```

Output:

```
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1
Enter the element to be inserted: A

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1
Enter the element to be inserted: B
```

```
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1
Enter the element to be inserted: C

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1

Enter the element to be inserted: D
~~Circular Queue Overflow~~

~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 3

Circular Queue contents are:
Front[0]-> A B C <-[2]Rear
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 2
Deleted element from the queue is: A
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 3
Circular Queue contents are:
Front[1]-> B C <-[2]Rear
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 1
Enter the element to be inserted: E
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
Enter Your Choice: 3
Circular Queue contents are:
Front[1]-> B C E <-[0]Rear
~~Main Menu~~
==> 1. Insertion and Overflow Demo
==> 2. Deletion and Underflow Demo
==> 3. Display
==> 4. Exit
```

```
Enter Your Choice: 4
```

Lab Program 7:

Design, Develop and Implement a menu driven Program in C for the following operations on Singly Linked List (SLL) of Student Data with the fields: USN, Name, Branch, Sem, PhNo

a. Create a SLL of N Students Data by using front insertion.

b. Display the status of SLL and count the number of nodes in it

c. Perform Insertion / Deletion at End of SLL

d. Perform Insertion / Deletion at Front of SLL(Demonstration of stack)

e. Exit

```c
#include<stdio.h>
#include<stdlib.h>
struct node
{
    char usn[25],name[25],branch[25];
    int sem;
    long int phone;
    struct node *link;
};
typedef struct node * NODE;
NODE start = NULL;
int count=0;

NODE create()
{
    NODE snode;
    snode = (NODE)malloc(sizeof(struct node));

    if(snode == NULL)
    {
        printf("\nMemory is not available");
        exit(1);
    }
    printf("\nEnter the usn,Name,Branch, sem,PhoneNo of the student:");
    scanf("%s %s %s %d %ld",snode->usn, snode->name, snode->branch, &snode->sem, &snode-
>phone);
    snode->link=NULL;
    count++;
    return snode;
}
NODE insertfront()
{
    NODE temp;
    temp = create();
    if(start == NULL)
    {
        return temp;
    }
    temp->link = start;
    return temp;
}
NODE deletefront()
{
    NODE temp;
    if(start == NULL)
    {
        printf("\nLinked list is empty");
        return NULL;
    }
    if(start->link == NULL)
    {
            printf("\nThe Student node with usn:%s is deleted ",start->usn);
            count--;
            free(start);
            return NULL;
    }
    temp = start;
    start = start->link;
    printf("\nThe Student node with usn:%s is deleted",temp->usn);
    count--;
    free(temp);
    return start;
}
NODE insertend()
{
    NODE cur,temp;
    temp = create();

    if(start == NULL)
    {
      return temp;
    }
    cur = start;
    while(cur->link !=NULL)
    {
         cur = cur->link;
    }
    cur->link = temp;
    return start;
}
NODE deleteend()
{
     NODE cur,prev;
     if(start == NULL)
     {
        printf("\nLinked List is empty");
        return NULL;
     }

     if(start->link == NULL)
     {
        printf("\nThe student node with the usn:%s is deleted",start->usn);
        free(start);
        count--;
        return NULL;
     }
     prev = NULL;
     cur = start;
     while(cur->link!=NULL)
     {
         prev = cur;
         cur = cur->link;
     }
      printf("\nThe student node with the usn:%s is deleted",cur->usn);
      free(cur);
      prev->link = NULL;
      count--;
      return start;
}
void display()
{
    NODE cur;
    int num=1;
    if(start == NULL)
    {
        printf("\nNo Contents to display in SLL \n");
        return;
    }
    printf("\nThe contents of SLL: \n");
    cur = start;
    while(cur!=NULL)
    {
       printf("\n||%d||  USN:%s|  Name:%s|  Branch:%s|  Sem:%d|  Ph:%ld|",num,cur->usn,  cur-
>name,cur->branch, cur->sem,cur->phone);
       cur = cur->link;
       num++;
    }
    printf("\n No of student nodes is %d \n",count);
}
void stackdemo()
{
   int ch;
   while(1)
   {
     printf("\n~~~Stack Demo using SLL~~~\n");
     printf("\n1:Push operation \n2: Pop operation \n3: Display \n4:Exit \n");
     printf("\nEnter your choice for stack demo");
     scanf("%d",&ch);
     switch(ch)
     {
        case 1: start = insertfront();
                break;
        case 2: start = deletefront();
                break;
```

```
        case 3: display();
                break;
       default : return;
     }
   }
   return;
}
int main()
{
    int ch,i,n;
    while(1)
    {
        printf("\n~~~Menu~~~");
        printf("\nEnter your choice for SLL operation \n");
        printf("\n1:Create SLL of Student Nodes");
        printf("\n2:DisplayStatus");
        printf("\n3:InsertAtEnd");
        printf("\n4:DeleteAtEnd");
        printf("\n5:Stack Demo using SLL(Insertion and Deletion at Front)");
        printf("\n6:Exit \n");
        printf("\nEnter your choice:");
        scanf("%d",&ch);
        switch(ch)
        {
        case 1 : printf("\nEnter the no of students:    ");
                 scanf("%d",&n);
                 for(i=1;i<=n;i++)
                    start = insertfront();
                 break;
        case 2: display();
                break;
```

```
        case 3: start = insertend();
              break;
        case 4: start = deleteend();
              break;
        case 5: stackdemo();
              break;
        case 6: exit(0);
        default: printf("\nPlease enter the valid choice");
        }
    }
}
```

Output:

```
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:1
Enter the no of students:    3

Enter the usn,Name,Branch, sem,PhoneNo of the student:
111
aaa
cs
1
111111
```

```
Enter the usn,Name,Branch, sem,PhoneNo of the student:
222
bbb
ec
2
222222

Enter the usn,Name,Branch, sem,PhoneNo of the student:
333
ccc
ec
3
333333
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:2
The contents of SLL:

||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
 No of student nodes is 3
~~~Menu~~~
Enter your choice for SLL operation
```

```
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:3
Enter the usn,Name,Branch, sem,PhoneNo of the student:
444
ddd
ec
4
444444
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:2
The contents of SLL:

||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
||4|| USN:444| Name:ddd| Branch:ec| Sem:4| Ph:444444|
 No of student nodes is 4
~~~Menu~~~
Enter your choice for SLL operation
```

```
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:4
The student node with the usn: 444 is deleted
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:2
The contents of SLL:
||1|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||2|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
||3|| USN:111| Name:aaa| Branch:cs| Sem:1| Ph:111111|
 No of student nodes is 3


~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
```

```
6:Exit
Enter your choice:4
The student node with the usn: 111 is deleted
~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:5
~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 1
Enter the usn,Name,Branch, sem,PhoneNo of the student:
555
eee
cs
1
555555
~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo:3
The contents of SLL:
```

```
||1|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||2|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||3|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
 No of student nodes is 3
~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 1
Enter the usn,Name,Branch, sem,PhoneNo of the student:
666
fff
cs
6
666666
~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 3
The contents of SLL:

||1|| USN:666| Name:fff| Branch:cs| Sem:6| Ph:666666|
||2|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||3|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||4|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
 No of student nodes is 4
~~~Stack Demo using SLL~~~
1:Push operation
```

```
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 2

The Student node with usn: 666 is deleted

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 3

The contents of SLL:
||1|| USN:555| Name:eee| Branch:cs| Sem:1| Ph:555555|
||2|| USN:333| Name:ccc| Branch:ec| Sem:3| Ph:333333|
||3|| USN:222| Name:bbb| Branch:ec| Sem:2| Ph:222222|
 No of student nodes is 3

~~~Stack Demo using SLL~~~
1:Push operation
2: Pop operation
3: Display
4:Exit
Enter your choice for stack demo: 4

~~~Menu~~~
Enter your choice for SLL operation
1:Create SLL of Student Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:Stack Demo using SLL(Insertion and Deletion at Front)
6:Exit
Enter your choice:6
```

Lab program 8:

Design, Develop and Implement a menu driven Program in C for the following operations on Doubly Linked List (DLL) of Employee Data with the fields: SSN, Name, Dept, Designation, Sal, PhNo

a. Create a DLL of N Employees Data by using end insertion.

b. Display the status of DLL and count the number of nodes in it

c. Perform Insertion and Deletion at End of DLL

d. Perform Insertion and Deletion at Front of DLL

e. Demonstrate how this DLL can be used as Double Ended Queue

f. Exit

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

struct node
{
        char ssn[25],name[25],dept[10],designation[25];
        int sal;
        long int phone;
        struct node *llink;
        struct node *rlink;
};
typedef struct node* NODE;

NODE first = NULL;
int count=0;



NODE create()
{
         NODE enode;
         enode = (NODE)malloc(sizeof(struct node));
         if( enode== NULL)
         {
                 printf("\nRunning out of memory");
                 exit(0);
          }
          printf("\nEnter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee: 
\n");
```

```
        scanf("%s  %s  %s  %s  %d  %ld", enode->ssn,  enode->name,  enode->dept,  enode->designation, &enode->sal, &enode->phone);
         enode->llink=NULL;
         enode->rlink=NULL;
         count++;
         return enode;
}

NODE insertfront()
{
        NODE temp;
        temp = create();
         if(first == NULL)
        {
               return temp;
         }
         temp->rlink = first;
         first->llink = temp;
         return temp;
}

void display()
{
         NODE cur;
         int nodeno=1;
         cur = first;
         if(cur == NULL)
                   printf("\nNo Contents to display in DLL");
         while(cur!=NULL)
        {
```

```
        printf("\nENode:%d||SSN:%s|Name:%s|Department:%s|Designation:%s|Salary:%d|Phone
no:%ld", nodeno, cur->ssn, cur->name,cur->dept, cur->designation, cur->sal, cur->phone);
                cur = cur->rlink;
                 nodeno++;
        }
         printf("\nNo of employee nodes is %d",count);
}

NODE deletefront()
{
        NODE temp;
        if(first == NULL)
        {
                printf("\nDoubly Linked List is empty");
                return NULL;
        }
        if(first->rlink== NULL)
       {
                printf("\nThe employee node with the ssn:%s is deleted", first->ssn);
               free(first);
               count--;
                return NULL;
        }
        temp = first;
        first = first->rlink;
        temp->rlink = NULL;
        first->llink = NULL;
        printf("\nThe employee node with the ssn:%s is deleted",temp->ssn);
        free(temp);
        count--;
        return first;
```

```
}

NODE insertend()
{
     NODE cur, temp;
     temp = create();

     if(first == NULL)
     {
          return temp;
     }
    cur= first;
    while(cur->rlink!=NULL)
    {
         cur = cur->rlink;
    }

    cur->rlink = temp;
    temp->llink = cur;
     return first;
}

NODE deleteend()
{
    NODE prev,cur;
    if(first == NULL)
    {
        printf("\nDoubly Linked List is empty");
        return NULL;
    }
```

```
        if(first->rlink == NULL)
        {
                printf("\nThe employee node with the ssn:%s is deleted",first->ssn);
                free(first);
                count--;
                return NULL;
        }

         prev=NULL;
         cur=first;

        while(cur->rlink!=NULL)
        {
               prev=cur;
               cur = cur->rlink;
         }

         cur->llink = NULL;
         printf("\nThe employee node with the ssn:%s is deleted",cur->ssn);
         free(cur);
         prev->rlink = NULL;
         count--;
         return first;
}

void deqdemo()
{
      int ch;
      while(1)
     {
           printf("\nDemo Double Ended Queue Operation");
     printf("\n1:InsertQueueFront\n          2:          DeleteQueueFront\n          3:InsertQueueRear\n
4:DeleteQueueRear\n 5:DisplayStatus\n 6: Exit \n");
             scanf("%d", &ch);

             switch(ch)
             {
                    case 1: first=insertfront();
                                break;
                   case 2: first=deletefront();
                              break;
                  case 3: first=insertend();
                              break;
                  case 4: first=deleteend();
                             break;
                   case 5: display();
                              break;
                  default : return;
           }
     }
}

void main()
{
    int ch,i,n;
    while(1)
    {
        printf("\n\n~~~Menu~~~");
        printf("\n1:Create DLL of Employee Nodes");
        printf("\n2:DisplayStatus");
         printf("\n3:InsertAtEnd");
         printf("\n4:DeleteAtEnd");
```

```
 printf("\n5:InsertAtFront");
 printf("\n6:DeleteAtFront");
 printf("\n7:Double Ended Queue Demo using DLL");
 printf("\n8:Exit \n");
 printf("\nPlease enter your choice: ");
scanf("%d",&ch);

switch(ch)
{
 case 1 : printf("\nEnter the no of Employees:   ");
          scanf("%d",&n);
          for(i=1;i<=n;i++)
          first = insertend();
          break;

 case 2:  display();
          break;

  case 3: first = insertend();
         break;

  case 4: first = deleteend();
         break;

  case 5: first = insertfront();
         break;

  case 6: first = deletefront();
        break;

  case 7: deqdemo();
```

```
                break;

            case 8 : exit(0);
          default: printf("\nPlease Enter the valid choice");
          }
    }
}
```

Output:

```
~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 1
Enter the no of Employees:   2

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
111
aaa
dept1
des1
1000
11111

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
222
```

```
bbb
dept2
des2
2000
22222

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 2

ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone
no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone
no:22222
No of employee nodes is 2

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
```

```
8:Exit
Please enter your choice: 3

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
333
ccc
dept3
des3
3000
33333

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 2

ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone
no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone
no:22222
ENode:3||SSN:333|Name:ccc|Department:dept3|Designation:des3|Salary:3000|Phone
no:33333
No of employee nodes is 3

~~~Menu~~~
```

```
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 5

Enter the ssn,Name,Department,Designation,Salary,PhoneNo of the employee:
444
ddd
dept4
des4
4000
44444

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 2
```

```
ENode:1||SSN:444|Name:ddd|Department:dept4|Designation:des4|Salary:4000|Phone
no:44444
ENode:2||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone
no:11111
ENode:3||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone
no:22222
ENode:4||SSN:333|Name:ccc|Department:dept3|Designation:des3|Salary:3000|Phone
no:33333
No of employee nodes is 4

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 4

The employee node with the ssn:333 is deleted

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
```

```
8:Exit
Please enter your choice: 6

The employee node with the ssn:444 is deleted

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
Please enter your choice: 2

ENode:1||SSN:111|Name:aaa|Department:dept1|Designation:des1|Salary:1000|Phone
no:11111
ENode:2||SSN:222|Name:bbb|Department:dept2|Designation:des2|Salary:2000|Phone
no:22222
No of employee nodes is 2

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit
```

```
Please enter your choice: 7

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
2

The employee node with the ssn:111 is deleted

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
4

The employee node with the ssn:222 is deleted

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
```

```
2

Doubly Linked List is empty

Demo Double Ended Queue Operation
1:InsertQueueFront
2: DeleteQueueFront
3:InsertQueueRear
4:DeleteQueueRear
5:DisplayStatus
6: Exit
6

~~~Menu~~~
1:Create DLL of Employee Nodes
2:DisplayStatus
3:InsertAtEnd
4:DeleteAtEnd
5:InsertAtFront
6:DeleteAtFront
7:Double Ended Queue Demo using DLL
8:Exit

Please enter your choice: 8
```

Lab Program 9:

Design, Develop and Implement a menu driven Program in C for the following operations on Binary Search Tree (BST) of Integers

a. Create a BST of N Integers: 6, 9, 5, 2, 8, 15, 24, 14, 7, 8, 5, 2

b. Traverse the BST in Inorder, Preorder and Post Order

c. Search the BST for a given element (KEY) and report the appropriate messaged.

d. Exit

```
#include<stdio.h>
#include<stdlib.h>
struct BST
{
        int data;
        struct BST *lchild;
        struct BST *rchild;
};
typedef struct BST * NODE;
NODE create()
{
        NODE temp;
        temp = (NODE) malloc(sizeof(struct BST));
        printf("\nEnter The value: ");
        scanf("%d", &temp->data);


        temp->lchild = NULL;
        temp->rchild = NULL;
        return temp;
}
void insert(NODE root, NODE newnode);
void inorder(NODE root);
void preorder(NODE root);
void postorder(NODE root);
void search(NODE root);
void insert(NODE root, NODE newnode)
{
   /*Note: if newnode->data ==  root->data it will be skipped. No duplicate nodes are allowed */
                if (newnode->data < root->data)
        {
                if (root->lchild == NULL)
                        root->lchild = newnode;
                else
                        insert(root->lchild, newnode);
        }
        if (newnode->data > root->data)
        {
                if (root->rchild == NULL)
                        root->rchild = newnode;
                else
                        insert(root->rchild, newnode);
        }
}
void search(NODE root)
{
        int key;
        NODE cur;
        if(root == NULL)
        {
                printf("\nBST is empty.");
                return;
        }
        printf("\nEnter Element to be searched: ");
        scanf("%d", &key);
        cur = root;
        while (cur != NULL)
        {
                if (cur->data == key)
                {
                        printf("\nKey element is present in BST");
                return;
            }
            if (key < cur->data)
                cur = cur->lchild;
            else
                cur = cur->rchild;
        }
        printf("\nKey element is not found in the BST");
}
void inorder(NODE root)
{
        if(root != NULL)
        {
            inorder(root->lchild);
            printf("%d ", root->data);
            inorder(root->rchild);
        }
}
void preorder(NODE root)
{
        if (root != NULL)
        {
            printf("%d ", root->data);
            preorder(root->lchild);
            preorder(root->rchild);
        }
}
void postorder(NODE root)
{
        if (root != NULL)
        {
                postorder(root->lchild);
                postorder(root->rchild);
                printf("%d ", root->data);
        }
}
void main()
{
        int ch, key, val, i, n;
        NODE root = NULL, newnode;
        while(1)
        {
                printf("\n~~~~BST MENU~~~~");
                printf("\n1.Create a BST");
                printf("\n2.Search");
                printf("\n3.BST Traversals: ");
                printf("\n4.Exit");
                printf("\nEnter your choice: ");
                scanf("%d", &ch);
                switch(ch)
                {
                        case 1:         printf("\nEnter the number of elements: ");
                                        scanf("%d", &n);
                                        for(i=1;i<=n;i++)
                                        {
                                                newnode = create();
                                                if (root == NULL)
                                                        root = newnode;
                                                else
                                                        insert(root, newnode);
                                        }
                                        break;
```

```
                    case 2:          if (root == NULL)
                                           printf("\nTree Is Not Created");
                                     else
                                     {
                                          printf("\nThe Preorder display : ");
                                          preorder(root);
                                          printf("\nThe Inorder display : ");
                                          inorder(root);
                                          printf("\nThe Postorder display : ");
                                          postorder(root);
                                     }

                                     break;
                    case 3:          search(root);
                                     break;

                   case 4:    exit(0);
               }
          }
}
```

Output:

```
~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 1
Enter the number of elements: 12
Enter The value: 6
Enter The value: 9
Enter The value: 5
```

```
Enter The value: 2
Enter The value: 8
Enter The value: 15
Enter The value: 24
Enter The value: 14
Enter The value: 7
Enter The value: 8
Enter The value: 5
Enter The value: 2
~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 3

The Preorder display:     6    5    2    9    8    7    15   14   24
The Inorder display:      2    5    6    7    8    9    14   15   24
The Postorder display:    2    5    7    8    14   24   15   9    6

~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 2
Enter Element to be searched: 66
Key element is not found in the BST
~~~~BST MENU~~~~
1.Create a BST
2.Search
```

```
3.BST Traversals:
4.Exit
Enter your choice: 2
Enter Element to be searched: 14
Key element is present in BST
~~~~BST MENU~~~~
1.Create a BST
2.Search
3.BST Traversals:
4.Exit
Enter your choice: 4
```

Lab Program 10:

- **The *Create operation* creates the AVL Tree.**
- **The *Insert operation* adds the new element to the constructed AVL Tree.**
- **The *Delete operation* deletes the element from the constructed AVL Tree.**
- **The *Print operation* prints the Pre-order and In-order sequence of constructed AVL Tree along with the Balance Factor (BF) for each node in the AVL Tree.**

Test the Program for the following scenarios:

1] Construct the AVL Tree for the data = 10 15 9 12 13 79 45 36 22

2] Insert the new element 17 to the constructed AVL Tree.

3] Delete element 45 from the constructed AVL Tree.

4] Delete the Root element 13 from the constructed AVL Tree.

Menu-driven Program for the AVL Tree

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

```
typedef struct node
{  int data;
   struct node *left,*right;
   int ht;
}node;
  node *insert(node *,int);
  node *Delete(node *,int);
  void  preorder(node *);
  void  inorder(node *);
  int   height( node *);
  node *rotateright(node *);
  node *rotateleft(node *);
  node *RR(node *);
  node *LL(node *);
  node *LR(node *);
  node *RL(node *);
  int BF(node *);
void main()
{
    node *root=NULL;
    int x,n,i,op;
    do
        {
            printf("\n");
            printf("\n1) Create the AVL Tree");
            printf("\n2) Insert Element into the AVL Tree");
            printf("\n3) Delete Element from the AVL Tree ");
            printf("\n4) Print the AVL Tree");
            printf("\n5) Quit");
            printf("\nEnter Your Choice: ");
            scanf("%d",&op);
            switch(op)
                {
                case 1:printf("\nEnter Total Number of Elements in the AVL Tree: ");
                       scanf("%d",&n);
                       printf("\n Enter AVL Tree Elements: ");
                       root=NULL;
                       for(i=0;i<n;i++)
                       {
                        scanf("%d",&x);
                        root=insert(root,x);
                       }
                       break;
                case 2:printf("\nEnter a Element to Insert in the AVL Tree: ");
                       scanf("%d",&x);
                       root=insert(root,x);
                       break;
                case 3:printf("\nEnter a Element to Delete from the AVL Tree: ");
                       scanf("%d",&x);
                       root=Delete(root,x);
                       break;
                case 4:    printf("\nPreorder Sequence of the AVL Tree:\n");
                    preorder(root);
                    printf("\nInorder sequence of the AVL Tree:\n");
                    inorder(root);
                    break;
                 }
    }while(op!=5);
}
node * insert(node *T,int x)
{
    if(T==NULL)
    {
        T=(node*)malloc(sizeof(node));
        T->data=x;
        T->left=NULL;
        T->right=NULL;
    }
    else
        if(x > T->data)                // Insert in Right Subtree
        {
            T->right=insert(T->right,x);
            if(BF(T)==-2)
                if(x>T->right->data)
                    T=RR(T);
                else
                    T=RL(T);
        }
        else
            if(x<T->data)
            {
                T->left=insert(T->left,x);
                if(BF(T)==2)
                    if(x < T->left->data)
                        T=LL(T);
                    else
                        T=LR(T);
            }
            T->ht=height(T);
            return(T);
}
node * Delete(node *T,int x)
```

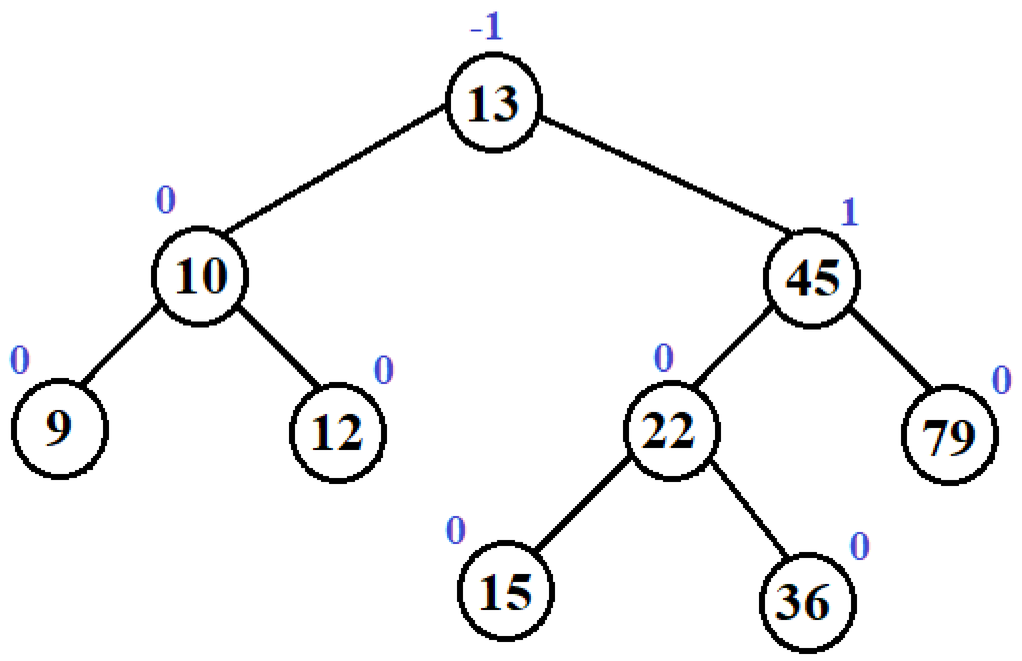
```
{      node *p;

   if(T==NULL)
   {
       return NULL;
   }
   else

       if(x > T->data)                // Insert in Right Subtree
       {
           T->right=Delete(T->right,x);
           if(BF(T)==2)
               if(BF(T->left)>=0)
                   T=LL(T);
               else
                   T=LR(T);
       }
       else
           if(x<T->data)
               {
                   T->left=Delete(T->left,x);
                   if(BF(T)==-2)              // Rebalance during windup
                       if(BF(T->right)<=0)
                           T=RR(T);
                       else
                           T=RL(T);
               }
           else
             {
               // Element to be deleted is found
                 if(T->right !=NULL)
```

```
                  {  // Delete its inorder succesor
                      p=T->right;
                      while(p->left != NULL)
                      p=p->left;

                      T->data=p->data;
                      T->right=Delete(T->right,p->data);
                      if(BF(T)==2)               // Rebalance during windup
                        if(BF(T->left)>=0)
                            T=LL(T);
                        else
                            T=LR(T);
                   }
                  else
                   return(T->left);

              }
    T->ht=height(T);
    return(T);
}
int height(node *T)
{
    int lh,rh;
    if(T==NULL)
        return(0);
    if(T->left==NULL)
        lh=0;
    else
        lh=1+T->left->ht;
    if(T->right==NULL)
        rh=0;
    else
        rh=1+T->right->ht;
    if(lh>rh)
        return(lh);
    return(rh);
}

node * rotateright(node *x)
{
    node *y;
    y=x->left;
    x->left=y->right;
    y->right=x;
    x->ht=height(x);
    y->ht=height(y);
    return(y);
}

node * rotateleft(node *x)
{
    node *y;
    y=x->right;
    x->right=y->left;
    y->left=x;
    x->ht=height(x);
    y->ht=height(y);
    return(y);
}

node * RR(node *T)
{
    T=rotateleft(T);
    return(T);
}

node * LL(node *T)
{
    T=rotateright(T);
    return(T);
}

node * LR(node *T)
{
    T->left=rotateleft(T->left);
    T=rotateright(T);
    return(T);
}

node * RL(node *T)
{
    T->right=rotateright(T->right);
    T=rotateleft(T);
    return(T);
}

int BF(node *T)
{
    int lh,rh;
    if(T==NULL)
    return(0);
    if(T->left==NULL)
        lh=0;
    else
        lh=1+T->left->ht;
    if(T->right==NULL)
        rh=0;
    else
        rh=1+T->right->ht;
    return(lh-rh);
}

void preorder(node *T)
{
    if(T!=NULL)
    {
        printf(" %d(Bf=%d)",T->data,BF(T));
        preorder(T->left);
        preorder(T->right);
    }
}

void inorder(node *T)
{
    if(T!=NULL)
    {
        inorder(T->left);
        printf(" %d(Bf=%d)",T->data,BF(T));
        inorder(T->right);
    }
}
```

OUTPUT -

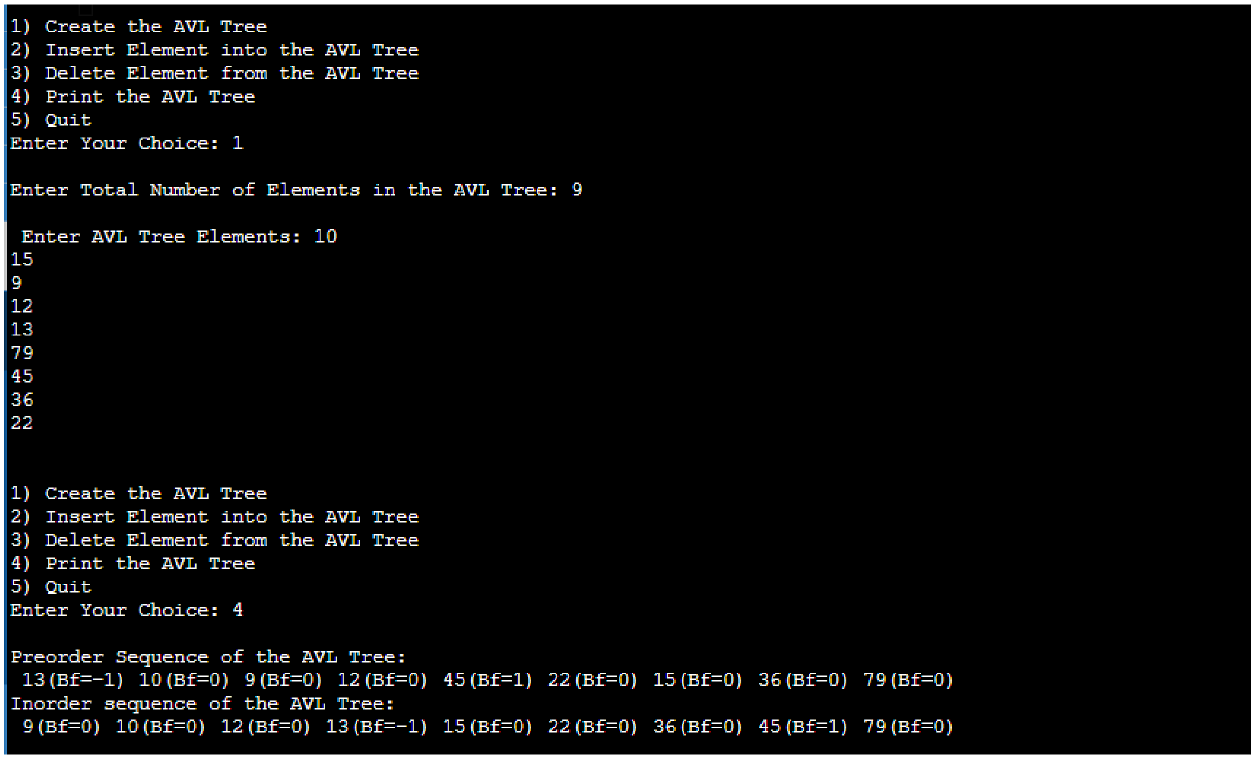
1] Construction of the AVL Tree for the data = 10 15 9 12 13 79 45 36 22

The AVL Tree for the above data looks as follows:



AVL Tree

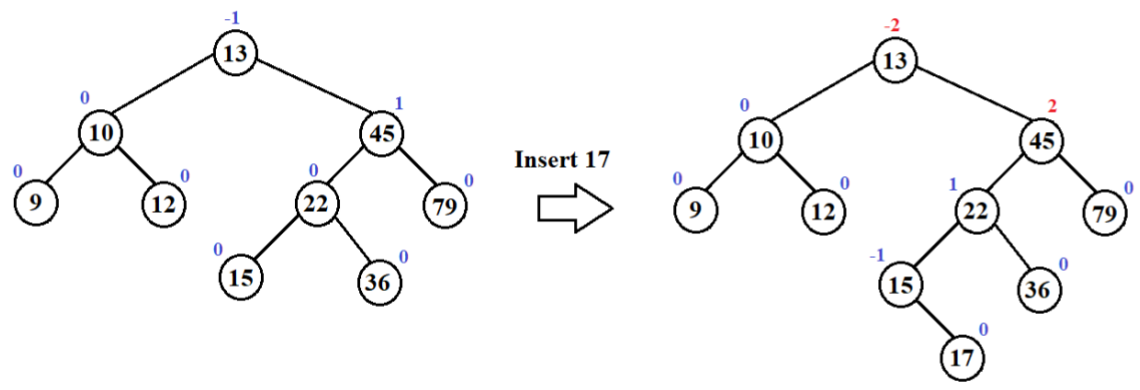
The program shows the output of AVL Tree creation as follows:

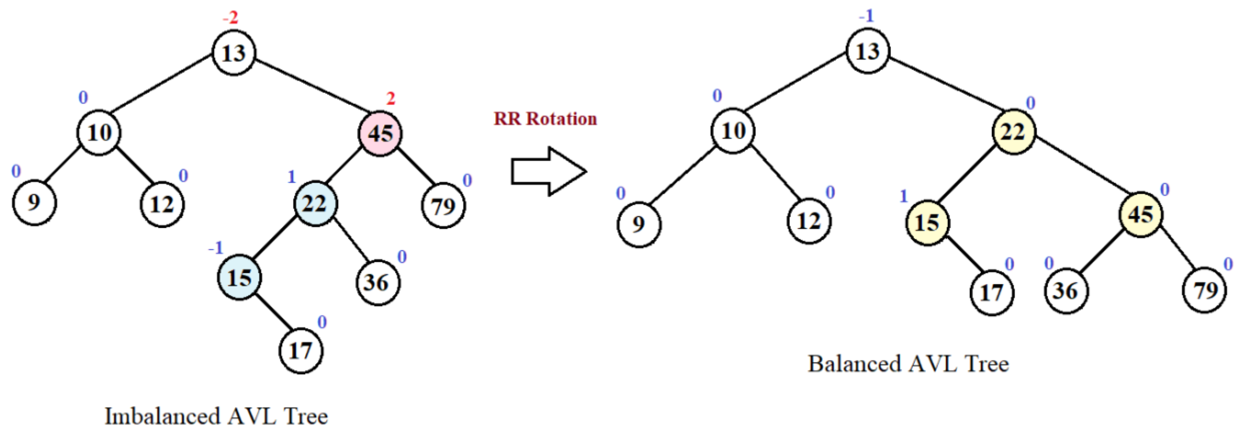


```
1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 1

Enter Total Number of Elements in the AVL Tree: 9

 Enter AVL Tree Elements: 10
15
9
12
13
79
45
36
22


1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 4

Preorder Sequence of the AVL Tree:
 13(Bf=-1) 10(Bf=0) 9(Bf=0) 12(Bf=0) 45(Bf=1) 22(Bf=0) 15(Bf=0) 36(Bf=0) 79(Bf=0)
Inorder sequence of the AVL Tree:
 9(Bf=0) 10(Bf=0) 12(Bf=0) 13(Bf=-1) 15(Bf=0) 22(Bf=0) 36(Bf=0) 45(Bf=1) 79(Bf=0)
```

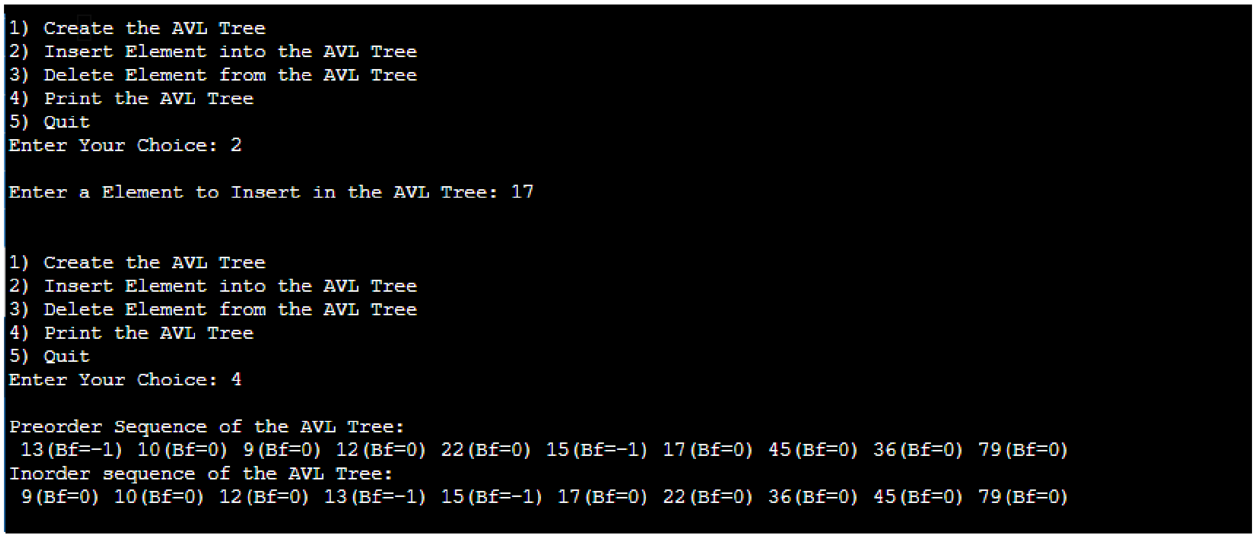
2] Insert the new element 17 to the constructed AVL Tree.

Insertion of the new element 17 to the constructed AVL Tree is done as follows;





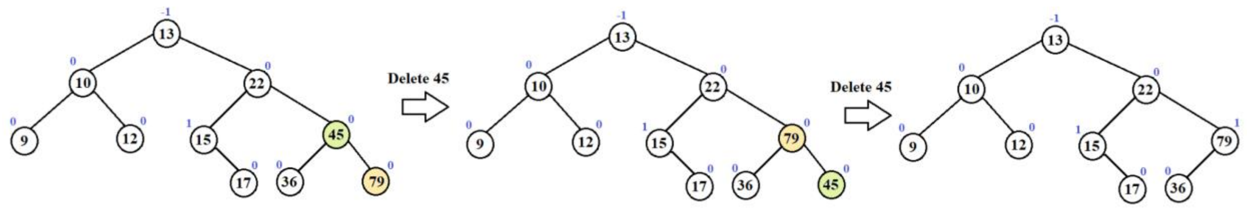
The program shows the output of AVL Tree insertion as follows:



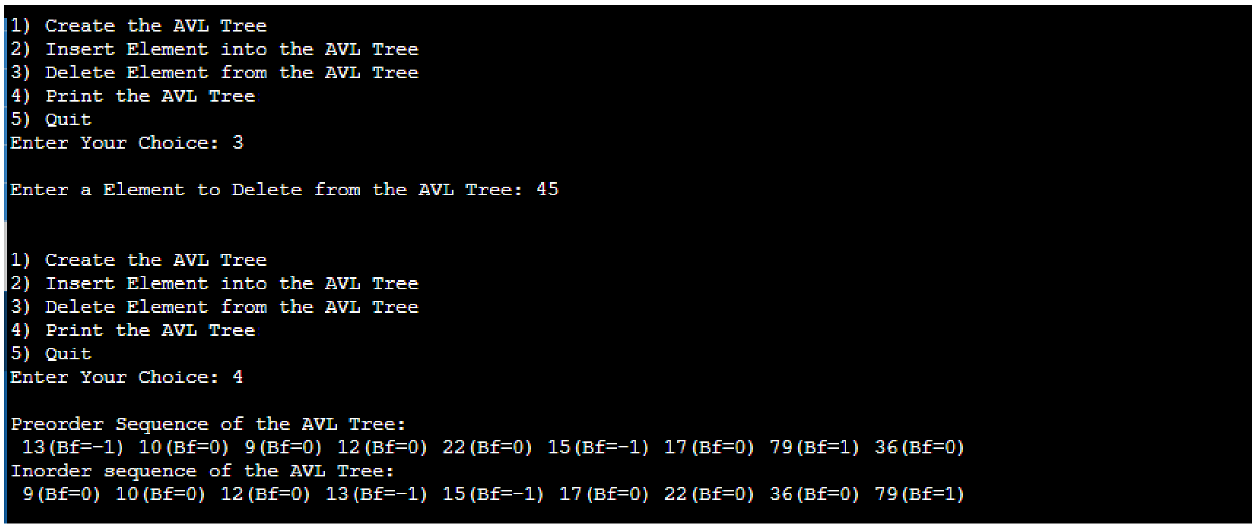
```
1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 2

Enter a Element to Insert in the AVL Tree: 17


1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 4

Preorder Sequence of the AVL Tree:
 13(Bf=-1) 10(Bf=0) 9(Bf=0) 12(Bf=0) 22(Bf=0) 15(Bf=-1) 17(Bf=0) 45(Bf=0) 36(Bf=0) 79(Bf=0)
Inorder sequence of the AVL Tree:
 9(Bf=0) 10(Bf=0) 12(Bf=0) 13(Bf=-1) 15(Bf=-1) 17(Bf=0) 22(Bf=0) 36(Bf=0) 45(Bf=0) 79(Bf=0)
```

3] Delete element 45 from the constructed AVL Tree.

Deletion of element 45 from the constructed AVL Tree is done as follows;



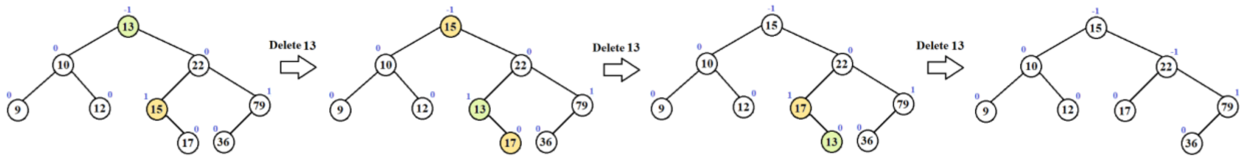
The program shows the output of AVL Tree Deletion as follows:



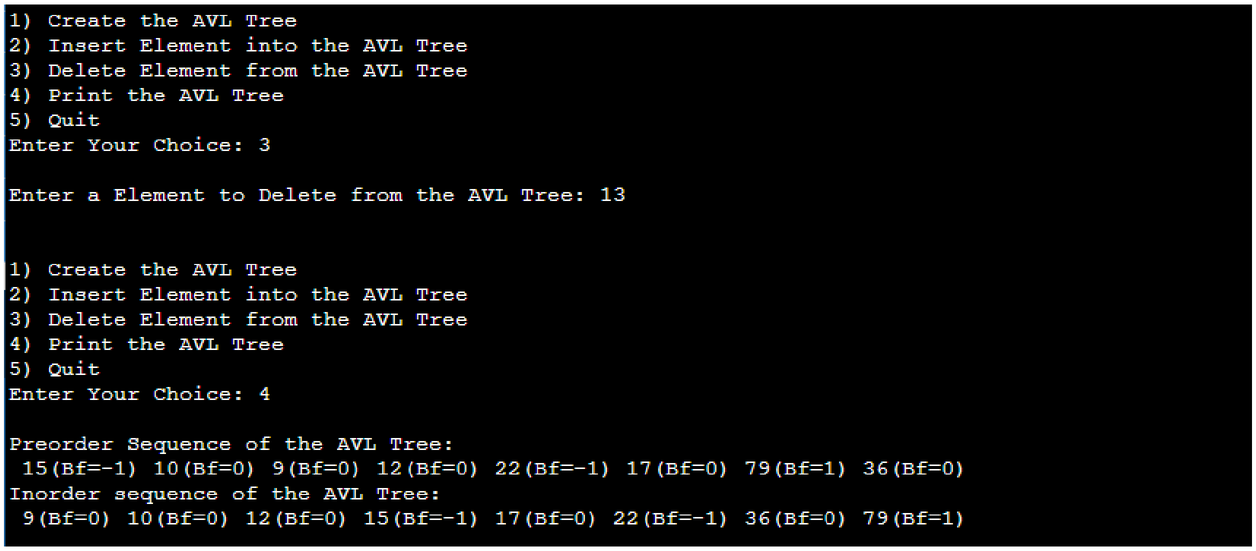
```
1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 3

Enter a Element to Delete from the AVL Tree: 45


1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 4

Preorder Sequence of the AVL Tree:
 13(Bf=-1) 10(Bf=0) 9(Bf=0) 12(Bf=0) 22(Bf=0) 15(Bf=-1) 17(Bf=0) 79(Bf=1) 36(Bf=0)
Inorder sequence of the AVL Tree:
 9(Bf=0) 10(Bf=0) 12(Bf=0) 13(Bf=-1) 15(Bf=-1) 17(Bf=0) 22(Bf=0) 36(Bf=0) 79(Bf=1)
```

4] Delete the Root element 13 from the constructed AVL Tree.

Deletion of Root element 13 from the constructed AVL Tree is done as follows;



The program shows the output of AVL Tree Deletion as follows:


```
1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 3

Enter a Element to Delete from the AVL Tree: 13


1) Create the AVL Tree
2) Insert Element into the AVL Tree
3) Delete Element from the AVL Tree
4) Print the AVL Tree
5) Quit
Enter Your Choice: 4

Preorder Sequence of the AVL Tree:
 15(Bf=-1) 10(Bf=0) 9(Bf=0) 12(Bf=0) 22(Bf=-1) 17(Bf=0) 79(Bf=1) 36(Bf=0)
Inorder sequence of the AVL Tree:
 9(Bf=0) 10(Bf=0) 12(Bf=0) 15(Bf=-1) 17(Bf=0) 22(Bf=-1) 36(Bf=0) 79(Bf=1)
```