Practical No:-01

Aim: Write a program in C++ to implement Array.

Solution:-

```
#include <iostream>

using namespace std;

int main()

{

float percentage[] = {56.4 , 99.0, 12.20, 67.2};

cout<<"printing all values of the array :\n";

for(int i = 0; i<4 ; i++){

cout<<"element "<<i+1<<" = "<<percentage[i]<<endl;

}

return 0;

}
```

Practical No:-02

Aim:- Write a program to accept the elements in 2D array and perform all the matrix operations i.e. addition, multiplication, transpose etc.

Solution:-

Addition of Matrix

```
#include <iostream>

using namespace std;

int main() {

int r=2, c=4, sum[2][4], i, j;

int a[2][4] = {{1,5,9,4} , {3,2,8,3}};

int b[2][4] = {{6,3,8,2} , {1,5,2,9}};

cout<<"The first matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<a[i][j]<<" ";

cout<<endl;

}

cout<<endl;

cout<<"The second matrix is: "<<endl;

for(i=0; i<r; ++i) {

for(j=0; j<c; ++j)

cout<<b[i][j]<<" ";

cout<<endl;

}

cout<<endl;

for(i=0;i<r;++i)

for(j=0;j<c;++j)

sum[i][j]=a[i][j]+b[i][j];

cout<<"Sum of the two matrices is:"<<endl;

for(i=0; i<r; ++i) {
```

```
for(j=0; j<c; ++j)

cout<<sum[i][j]<<" ";

cout<<endl;

}

return 0;

}
```

Solution:-

Subtraction of Matrix

```
#include <iostream>
using namespace std;
int main() {
int r=2, c=4, sub[2][4], i, j;
int a[2][4] = {{1,5,9,4} , {3,2,8,3}};
int b[2][4] = {{6,3,8,2} , {1,5,2,9}};
cout<<"The first matrix is: "<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<a[i][j]<<" ";
cout<<endl;
}
cout<<endl;
cout<<"The second matrix is: "<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<b[i][j]<<" ";
cout<<endl;
}
cout<<endl;
for(i=0;i<r;++i)
for(j=0;j<c;++j)
sub[i][j]=a[i][j]-b[i][j];
cout<<"Subtraction of the two matrices
is:"<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<sub[i][j]<<" ";
cout<<endl;
}
return 0;
}
```

Solution:-

Multiplication of Matrix

```
#include <iostream>
using namespace std;
int main() {
int r=2, c=4, mul[2][4], i, j;
int a[2][4] = {{1,5,9,4} , {3,2,8,3}};
int b[2][4] = {{6,3,8,2} , {1,5,2,9}};
cout<<"The first matrix is: "<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<a[i][j]<<" ";
cout<<endl;
}
cout<<endl;
cout<<"The second matrix is: "<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<b[i][j]<<" ";
cout<<endl;
}
cout<<endl;
for(i=0;i<r;++i)
for(j=0;j<c;++j)
mul[i][j]=a[i][j]*b[i][j];
cout<<"mul of the two matrices
is:"<<endl;
for(i=0; i<r; ++i) {
for(j=0; j<c; ++j)
cout<<mul[i][j]<<" ";
cout<<endl;
}
return 0;
}
```

Practical No:-03

Aim: - **Explain following techniques**
Technique Bubble sort

```
#include<iostream>
using namespace std;
void swapping(int &a, int &b)
{ //swap the content of a and b
int temp;
temp = a;
a = b;
b = temp;
}
void display(int *array, int size)
{
for(int i = 0; i<size; i++)
cout << array[i] << " ";
cout << endl;
}
void bubbleSort(int *array, int size)
{
for(int i = 0; i<size; i++) {
int swaps = 0; //flag to detect any swap
is there or not
for(int j = 0; j<size-i-1; j++) {
if(array[j] > array[j+1]) { //when the
current item is bigger than next
swapping(array[j], array[j+1]);
swaps = 1; //set swap flag
}
}
if(!swaps)
break; // No swap in this pass, so array
is sorted
}
}
int main() {
int n;
cout << "Enter the number of elements: ";
cin >> n;
int arr[n]; //create an array with given
number of elements
cout << "Enter elements:" << endl;
for(int i = 0; i<n; i++) {
cin >> arr[i];
}
cout << "Array before Sorting: ";
display(arr, n);
bubbleSort(arr, n);
cout << "Array after Sorting: ";
display(arr, n);
}
```

Solution 2:- C++ program for insertion sort

```
#include <bits/stdc++.h>
using namespace std;

// Function to sort an array using
// insertion sort
void insertionSort(int arr[], int n)
{
int i, key, j;
for (i = 1; i < n; i++)
{
key = arr[i];
j = i - 1;

// Move elements of arr[0..i-1],
// that are greater than key, to one
// position ahead of their
// current position
while (j >= 0 && arr[j] > key)
{
arr[j + 1] = arr[j];
j = j - 1;
}
arr[j + 1] = key;
}
}

// A utility function to print an array
// of size n
void printArray(int arr[], int n)
{
int i;
for (i = 0; i < n; i++)
cout << arr[i] << " ";
cout << endl;
}

// Driver code
int main()
{
int arr[] = { 12, 11, 13, 5, 6 };
int N = sizeof(arr) / sizeof(arr[0]);

insertionSort(arr, N);
printArray(arr, N);
return 0;
}
```

Solution 3:- C++ implementation of Radix Sort

```
#include <iostream>
using namespace std;

// A utility function to get maximum
value in arr[]
int getMax(int arr[], int n)
{
int mx = arr[0];
for (int i = 1; i < n; i++)
if (arr[i] > mx)
mx = arr[i];
return mx;
}

// A function to do counting sort of
arr[] according to
// the digit represented by exp.
void countSort(int arr[], int n, int exp)
{
int output[n]; // output array
int i, count[10] = { 0 };

// Store count of occurrences in count[]
for (i = 0; i < n; i++)
count[(arr[i] / exp) % 10]++;

// Change count[i] so that count[i] now
contains actual
// position of this digit in output[]
for (i = 1; i < 10; i++)
count[i] += count[i - 1];
// Build the output array
for (i = n - 1; i >= 0; i--) {
output[count[(arr[i] / exp) % 10] - 1] =
arr[i];
count[(arr[i] / exp) % 10]--;
}

// Copy the output array to arr[], so
that arr[] now
// contains sorted numbers according to
current digit
for (i = 0; i < n; i++)
arr[i] = output[i];
}

// The main function to that sorts arr[]
of size n using
// Radix Sort
void radixsort(int arr[], int n)
{
// Find the maximum number to know number
of digits
int m = getMax(arr, n);

// Do counting sort for every digit. Note
that instead
// of passing digit number, exp is
passed. exp is 10^i
// where i is current digit number
for (int exp = 1; m / exp > 0; exp *= 10)
countSort(arr, n, exp);
}

// A utility function to print an array
void print(int arr[], int n)
{
for (int i = 0; i < n; i++)
cout << arr[i] << " ";
}

// Driver Code
int main()
{
int arr[] = { 170, 45, 75, 90, 802, 24,
2, 66 };
int n = sizeof(arr) / sizeof(arr[0]);

// Function Call
radixsort(arr, n);
print(arr, n);
return 0;
}
```

Aim: Suppose an array contains n elements. Given a number x that may occur several times in the array. Write a program to find
i. The number of occurrences of x in the array
ii. The position of first occurrence of x in the array.

Solution 1:- C++ program to count occurrences of an element

```
#include<bits/stdc++.h>
using namespace std;

// Returns number of times x occurs in
arr[0..n-1]
int countOccurrences(int arr[], int n,
int x)
{
int res = 0;
for (int i=0; i<n; i++)
if (x == arr[i])
res++;
return res;
}

// Driver code
int main()
{
int arr[] = {1, 2, 2, 2, 2, 3, 4, 7 ,8 ,8
};
int n = sizeof(arr)/sizeof(arr[0]);
int x = 2;
cout << countOccurrences(arr, n, x);
return 0;
}
```

Solution 2:- C++ program to count occurrences of an element in a sorted array.

```
# include <bits/stdc++.h>
using namespace std;

/* if x is present in arr[] then returns
the count
      of occurrences of x, otherwise
returns 0. */
int count(int arr[], int x, int n)
{
/* get the index of first occurrence of x
*/
int *low = lower_bound(arr, arr+n, x);

// If element is not present, return 0
if (low == (arr + n) || *low != x)
      return 0;

/* Else get the index of last occurrence
of x.
      Note that we are only looking in
the
      subarray after first occurrence */
int *high = upper_bound(low, arr+n, x);

/* return count */
return high - low;
}

/* driver program to test above functions
*/
int main()
{
int arr[] = {1, 2, 2, 3, 3, 3, 3};
int x = 3; // Element to be counted in
arr[]
int n = sizeof(arr)/sizeof(arr[0]);
int c = count(arr, x, n);
printf(" %d occurs %d times ", x, c);
return 0;
}
```

Practical No: - 04

Aim: Write a program in C++ to delete particular element from an array of 10 integers.

```
#include<iostream>
using namespace std;
int main()
{
int arr[10], tot=10, i, elem, j, found=0;
cout<<"Enter 10 Array Elements: ";
for(i=0; i<tot; i++)
cin>>arr[i];
cout<<"\nEnter Element to Delete: ";
cin>>elem;
for(i=0; i<tot; i++)
{
if(arr[i]==elem)
{
for(j=i; j<(tot-1); j++)
arr[j] = arr[j+1];
found++;
i--;
tot--;
}
}
if(found==0)
cout<<"\nElement doesn't found in the
Array!";
else
cout<<"\nElement Deleted Successfully!";
cout<<endl;
return 0;
}
```

Practical No: - 05

Aim: - Consider two single dimensional array of size 20 and 3 respectively. Write a program in C++ to display all the elements which are common in both arrays.

```
#include <bits/stdc++.h>
using namespace std;
int main(){
//defining the array
int arr1[] = { 1, 45, 54, 71, 76, 12 };
int arr2[] = { 1, 7, 5, 4, 6, 12 };
int n1 = sizeof(arr1) / sizeof(arr1[0]);
int n2 = sizeof(arr2) / sizeof(arr2[0]);
sort(arr1, arr1 + n1);
sort(arr2, arr2 + n2);
cout << "First Array: ";
for (int i = 0; i < n1; i++)
cout << arr1[i] << " ";
cout << endl;
cout << "Second Array: ";
for (int i = 0; i < n2; i++)
cout << arr2[i] << " ";
cout << endl;
vector<int> v(n1 + n2);
vector<int>::iterator it, st;
//finding the common elements
it = set_intersection(arr1, arr1 + n1,
arr2, arr2 + n2, v.begin());
cout << "\nCommon elements:\n";
for (st = v.begin(); st != it; ++st)
cout << *st << ", ";
cout << '\n';
return 0;
}
```

Aim:- Write a program to build a sparse matrix as an array.

// C++ program for Sparse Matrix Representation using Array

```
#include <iostream>
using namespace std;

int main()
{
	// Assume 4x5 sparse matrix
	int sparseMatrix[4][5] =
	{
		{0 , 0 , 3 , 0 , 4 },
		{0 , 0 , 5 , 7 , 0 },
		{0 , 0 , 0 , 0 , 0 },
		{0 , 2 , 6 , 0 , 0 }
	};

	int size = 0;
	for (int i = 0; i < 4; i++)
		for (int j = 0; j < 5; j++)
			if (sparseMatrix[i][j]
!= 0)
				size++;

	// number of columns in
compactMatrix (size) must be
	// equal to number of non - zero
elements in
	// sparseMatrix
	int compactMatrix[3][size];

	// Making of new matrix
	int k = 0;
	for (int i = 0; i < 4; i++)
		for (int j = 0; j < 5; j++)
			if (sparseMatrix[i][j]
!= 0)
			{

	compactMatrix[0][k] = i;

	compactMatrix[1][k] = j;

	compactMatrix[2][k] =
sparseMatrix[i][j];
				k++;
			}

	for (int i=0; i<3; i++)
	{
		for (int j=0; j<size; j++)
				cout <<" "<<
compactMatrix[i][j];

			cout <<"\n";
	}
	return 0;
}
```

Write a menu driven program for stack contain following function

- **PUSH**
- **POP**
- **DISPLAY**
- **PEEK**

```
/*  C++ Menu Driven Program for Stack
Operations Using Arrays  */


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

using namespace std;

class Stack
{
int top;
int arr[50];
public:
 Stack()
{
 top=-1;
}

 void push();
 void pop();
 void view();
 int isEmpty();
 int isFull();
};

int Stack::isEmpty()
{
 return (top==(-1)?1:0);
}

int Stack::isFull()
{
 return ( top == 50 ? 1 : 0 );
}

void Stack::push()
{
 if(isFull())
 {
  cout<<"\nSTACK IS FULL { OVERFLOW }";
 }
 else
 {
  int i;
  cout<<"\nEnter an element :: ";
  cin>>i;
  ++top;
  arr[top]=i;
  cout<<"\nInsertion successful.\n";
 }
}

void Stack::pop()
{
 int num;
 if(isEmpty())
 {
 cout<<"\n STACK IS EMPTY [ UNDERFLOW ]
";
 }
 else
 {
 cout<<"\nDeleted item is :
"<<arr[top]<<"\n";
 top--;
 }
}

void Stack::view()
{
 if(isEmpty())
 {
 cout<<"\n STACK IS EMPTY [ UNDERFLOW ]
";
 }
 else
 {
 cout<<"\nSTACK :\n";
 for(int i=top;i>=0;i--)
 {
 cout<<arr[i]<<"\n";
 }
 }
}


int main()
{
 Stack s;
 int ch;
 ch=0;
 while(ch!=4)
 {
  cout<<"\n1. Push\n";
  cout<<"2. Pop\n";
  cout<<"3. Display\n";
  cout<<"4. Quit\n";
  cout<<"\nEnter your Choice :: ";
  cin>>ch;
```

```
  switch(ch)
  {
 case 1:
 s.push();
 break;

 case 2:
 s.pop();
 break;

 case 3:
 s.view();
 break;

 case 4:
 ch=4;
 cout<<"\nPress any key .. ";
 break;

 default:
 cout<<"\nWrong Choice!! \n";
 break;
  }
 }

 return 0;
}
```

Practical No: - 06

Aim: - Write a program in C++ to implement queue using Array.

```
#include <iostream>
using namespace std;
int queue[100], n = 100, front = - 1,
rear = - 1;
void Insert() {
int val;
if (rear == n - 1)
cout<<"Queue Overflow"<<endl;
else {
if (front == - 1)
front = 0;
cout<<"Insert the element in queue :
"<<endl;
cin>>val;
rear++;
queue[rear] = val;
}
}
void Delete() {
if (front == - 1 || front > rear) {
cout<<"Queue Underflow ";
return ;
} else {
cout<<"Element deleted from queue is :
"<< queue[front] <<endl;
front++;;
}
}
void Display() {
if (front == - 1)
cout<<"Queue is empty"<<endl;
else {
cout<<"Queue elements are : ";
for (int i = front; i <= rear; i++)
cout<<queue[i]<<" ";
cout<<endl;
}
}
int main() {
int ch;
cout<<"1) Insert element to queue"<<endl;
cout<<"2) Delete element from
queue"<<endl;
cout<<"3) Display all the elements of
queue"<<endl;
cout<<"4) Exit"<<endl;
do {
cout<<"Enter your choice : "<<endl;
cin>>ch;
switch (ch) {
case 1: Insert();
break;
case 2: Delete();
break;
case 3: Display();
break;
case 4: cout<<"Exit"<<endl;
break;
default: cout<<"Invalid choice"<<endl;
}
} while(ch!=4);
return 0;
}
```

Consider the single Linked List contains following elements: Rollno int, sname char(20),city char(20),course char(3) Write a program in C++ to represent linked List with the above elements.

```
// C++ program for the above approach
#include <bits/stdc++.h>
using namespace std;

// Node Class
class Node {
public:
 int roll;
 string Name;
 string Dept;
 int Marks;
 Node* next;
};

// Stores the head of the Linked List
Node* head = new Node();

// Check Function to check that if
// Record Already Exist or Not
bool check(int x)
{
 // Base Case
 if (head == NULL)
 return false;

 Node* t = new Node;
 t = head;

 // Traverse the Linked List
 while (t != NULL) {
 if (t->roll == x)
 return true;
 t = t->next;
 }

 return false;
}

// Function to insert the record
void Insert_Record(int roll, string Name,
  string Dept, int Marks)
{
 // if Record Already Exist
 if (check(roll)) {
 cout << "Student with this "
 << "record Already Exists\n";
 return;
 }

 // Create new Node to Insert Record
 Node* t = new Node();
 t->roll = roll;
 t->Name = Name;
 t->Dept = Dept;
 t->Marks = Marks;
 t->next = NULL;

 // Insert at Begin
 if (head == NULL
 || (head->roll >= t->roll)) {
 t->next = head;
 head = t;
 }

 // Insert at middle or End
 else {
 Node* c = head;
 while (c->next != NULL
 && c->next->roll < t->roll) {
 c = c->next;
 }
 t->next = c->next;
 c->next = t;
 }

 cout << "Record Inserted "
 << "Successfully\n";
}

// Function to search record for any
// students Record with roll number
void Search_Record(int roll)
{
 // if head is NULL
 if (!head) {
 cout << "No such Record "
 << "Available\n";
 return;
 }

 // Otherwise
 else {
 Node* p = head;
 while (p) {
 if (p->roll == roll) {
  cout << "Roll Number\t"
  << p->roll << endl;
  cout << "Name\t\t"
  << p->Name << endl;
  cout << "Department\t"
```

```
  << p->Dept << endl;
  cout << "Marks\t\t"
  << p->Marks << endl;
  return;
 }
 p = p->next;
 }

 if (p == NULL)
 cout << "No such Record "
  << "Available\n";
 }
}

// Function to delete record students
// record with given roll number
// if it exist
int Delete_Record(int roll)
{
 Node* t = head;
 Node* p = NULL;

 // Deletion at Begin
 if (t != NULL
 && t->roll == roll) {
 head = t->next;
 delete t;

 cout << "Record Deleted "
 << "Successfully\n";
 return 0;
 }

 // Deletion Other than Begin
 while (t != NULL && t->roll != roll) {
 p = t;
 t = t->next;
 }
 if (t == NULL) {
 cout << "Record does not Exist\n";
 return -1;
 p->next = t->next;

 delete t;
 cout << "Record Deleted "
 << "Successfully\n";

 return 0;
 }
}

// Function to display the Student's
// Record
void Show_Record()
{
 Node* p = head;
 if (p == NULL) {
 cout << "No Record "
 << "Available\n";
 }
 else {
 cout << "Index\tName\tCourse"
 << "\tMarks\n";

 // Until p is not NULL
 while (p != NULL) {
 cout << p->roll << " \t"
  << p->Name << "\t"
  << p->Dept << "\t"
  << p->Marks << endl;
 p = p->next;
 }
 }
}

// Driver code
int main()
{
 head = NULL;
 string Name, Course;
 int Roll, Marks;

 // Menu-driven program
 while (true) {
 cout << "\n\tWelcome to Student Record "
  "Management System\n\n\tPress\n\t1 to "
  "create a new Record\n\t2 to delete a "
  "student record\n\t3 to Search a
Student "
  "Record\n\t4 to view all students "
  "record\n\t5 to Exit\n";
 cout << "\nEnter your Choice\n";
 int Choice;

 // Enter Choice
 cin >> Choice;
 if (Choice == 1) {
 cout << "Enter Name of Student\n";
 cin >> Name;
 cout << "Enter Roll Number of
Student\n";
 cin >> Roll;
 cout << "Enter Course of Student \n";
 cin >> Course;
 cout << "Enter Total Marks of
Student\n";
```

```
 cin >> Marks;
 Insert_Record(Roll, Name, Course,
Marks);
 }
 else if (Choice == 2) {
 cout << "Enter Roll Number of Student
whose "
  "record is to be deleted\n";
 cin >> Roll;
 Delete_Record(Roll);
 }
 else if (Choice == 3) {
 cout << "Enter Roll Number of Student
whose "
  "record you want to Search\n";
 cin >> Roll;
 Search_Record(Roll);
 }
 else if (Choice == 4) {
 Show_Record();
 }
 else if (Choice == 5) {
 exit(0);
 }
 else {
 cout << "Invalid Choice "
  << "Try Again\n";
 }
 }
 return 0;
}
```

Practical No: - 07

Aim: - Write menu driven program which create and display the circular linked list

```
#include <bits/stdc++.h>
using namespace std;
struct Node { int data; struct Node
*next; }*Head=NULL;
void create(int a[],int n)
{
struct Node *t,*last;
Head=new Node;
Head->data=a[0];
Head->next=Head;
last=Head;
for(int i=1;i<n;i++) {
t=new Node;
t->data=a[i];
t->next=last->next;
last->next=t;
last=t;
}
}
int count(struct Node *p)
{
int c=0;
do
{
c++; p=p->next;
}
while(p!=Head);
return c;
}
void insert(int pos, int x)
{
if(pos>count(Head) || pos<0)
return;
struct Node *t,*p=Head;
t=new Node;
t->data=x;
if(pos==0)
{
if(Head==NULL)
{
Head=t;
Head->next=t;
}
while(p->next!=Head)
p=p->next;
t->next=Head;
p->next=t;
Head=t;
}
else{
for(int i=0;i<pos-1;i++)
p=p->next;
t->next=p->next;
p->next=t;
}
}
int Delete(struct Node *p,int index)
{
struct Node *q;
int x=-1;
if(index<1 || index>count(Head))
return x;
if(index==1)
{
x=p->data;
while(p->next!=Head)
p=p->next;
if(Head==p)
{
delete Head;
Head=NULL;
}
else
{
p->next=Head->next;
delete Head;
Head=p->next;
}
return x;
}
else
{
for(int i=0;i<index-2;i++)
{
p=p->next;
}
q=p->next;
p->next=q->next;
x=q->data;
delete q;
return x;
}
}
void display(struct Node *p)
{
do
{
cout<<p->data<<" ";
p=p->next;
}
```

```
while(p!=Head);
}
int main()
{
int a[500];
int option,n,pos,x,index,t;
do
{
cout<<"1. Create Circular Linked
list"<<endl;
cout<<"2. Insert in Circular Linked
list"<<endl;
cout<<"3. Delete "<<endl;
cout<<"4. Display"<<endl;
cout<<"5. Exit"<<endl;
cout<<"Enter an option :"<<endl;
cin>>option;
switch(option)
{
case 1:
{
cout<<"Enter no of integers : "<<endl;
cin>>n;
cout<<"Enter the numbers"<<endl;
for(int i=0;i<n;i++)
cin>>a[i];
create(a,n);
cout<<endl;
break;
}
case 2:
{
cout<<"Enter position to insert an
element : "<<endl;
cin>>pos;
cout<<"Enter element : "<<endl;
cin>>x;
insert(pos,x);
cout<<endl;
break;
}
case 3:
{
cout<<"Enter position to delete element :
"<<endl;
cin>>index;
cout<<"Deleted element is:
"<<Delete(Head,index);
cout<<endl;
break;
}
case 4:
{
cout<<"Displaying elements :";
display(Head);
cout<<endl;
break;
}
default:
cout<<"Exiting program......"<<endl;
}
}
while(option<=4);
return 0;
}
```

Practical No: - 08

Aim: - Create binary search tree 15, 2, 25, 45, 35, 23, 100, 5

```
# include <iostream>
# include <cstdlib>
using namespace std;
struct nod//node declaration
{
int info;
struct nod *l;
struct nod *r;
}*r;
class BST
{
public://functions declaration
void search(nod *, int);
void find(int, nod **, nod **);
void insert(nod *, nod *);
void del(int);
void casea(nod *,nod *);
void caseb(nod *,nod *);
void casec(nod *,nod *);
void preorder(nod *);
void inorder(nod *);
void postorder(nod *);
void show(nod *, int);
BST()
{
r = NULL;
}
};
void BST::find(int i, nod **par, nod
**loc)//find the position of the item
{
nod *ptr, *ptrsave;
if (r == NULL)
{
*loc = NULL;
*par = NULL;
return;
}
if (i == r->info)
{
*loc = r;
*par = NULL;
return;
}
if (i < r->info)
ptr = r->l;
else
ptr = r->r;
ptrsave = r;
while (ptr != NULL)
{
if (i == ptr->info)
{
*loc = ptr;
*par = ptrsave;
return;
}
ptrsave = ptr;
if (i < ptr->info)
ptr = ptr->l;
else
ptr = ptr->r;
}
*loc = NULL;
*par = ptrsave;
}
void BST::search(nod *root, int data)
//searching
{
int depth = 0;
nod *temp = new nod;
temp = root;
while(temp != NULL)
{
depth++;
if(temp->info == data)
{
cout<<"\nData found at depth:
"<<depth<<endl;
return;
}
else if(temp->info > data)
temp = temp->l;
else
temp = temp->r;
}
cout<<"\n Data not found"<<endl;
return;
}
void BST::insert(nod *tree, nod *newnode)
{
if (r == NULL)
{
r = new nod;
r->info = newnode->info;
r->l= NULL;
r->r= NULL;
cout<<"Root Node is Added"<<endl;
return;
}
if (tree->info == newnode->info)
```

```
{
cout<<"Element already in the
tree"<<endl;
return;
}
if (tree->info > newnode->info)
{
if (tree->l != NULL)
{
insert(tree->l, newnode);
}
else
{
tree->l= newnode;
(tree->l)->l = NULL;
(tree->l)->r= NULL;
cout<<"Node Added To Left"<<endl;
return;
}
}
else
{
if (tree->r != NULL)
{
insert(tree->r, newnode);
}
else
{
tree->r = newnode;
(tree->r)->l= NULL;
(tree->r)->r = NULL;
cout<<"Node Added To Right"<<endl;
return;
}
}
}
void BST::del(int i)
{
nod *par, *loc;
if (r == NULL)
{
cout<<"Tree empty"<<endl;
return;
}
find(i, &par, &loc);
if (loc == NULL)
{
cout<<"Item not present in tree"<<endl;
return;
}
if (loc->l == NULL && loc->r == NULL)
{
casea(par, loc);
cout<<"item deleted"<<endl;
}
if (loc->l!= NULL && loc->r == NULL)
{
caseb(par, loc);
cout<<"item deleted"<<endl;
}
if (loc->l== NULL && loc->r != NULL)
{
caseb(par, loc);
cout<<"item deleted"<<endl;
}
if (loc->l != NULL && loc->r != NULL)
{
casec(par, loc);
cout<<"item deleted"<<endl;
}
free(loc);
}
void BST::casea(nod *par, nod *loc )
{
if (par == NULL)
{
r= NULL;
}
else
{
if (loc == par->l)
par->l = NULL;
else
par->r = NULL;
}
}
void BST::caseb(nod *par, nod *loc)
{
nod *child;
if (loc->l!= NULL)
child = loc->l;
else
child = loc->r;
if (par == NULL)
{
r = child;
}
else
{
if (loc == par->l)
par->l = child;
else
par->r = child;
}
}
void BST::casec(nod *par, nod *loc)
```

```
{
nod *ptr, *ptrsave, *suc, *parsuc;
ptrsave = loc;
ptr = loc->r;
while (ptr->l!= NULL)
{
ptrsave = ptr;
ptr = ptr->l;
}
suc = ptr;
parsuc = ptrsave;
if (suc->l == NULL && suc->r == NULL)
casea(parsuc, suc);
else
caseb(parsuc, suc);
if (par == NULL)
{
r = suc;
}
else
{
if (loc == par->l)
par->l = suc;
else
par->r= suc;
}
suc->l = loc->l;
suc->r= loc->r;
}
void BST::preorder(nod *ptr)
{
if (r == NULL)
{
cout<<"Tree is empty"<<endl;
return;
}
if (ptr != NULL)
{
cout<<ptr->info<<" ";
preorder(ptr->l);
preorder(ptr->r);
}
}
void BST::inorder(nod *ptr)//inorder
traversal
{
if (r == NULL)
{
cout<<"Tree is empty"<<endl;
return;
}
if (ptr != NULL)
{
inorder(ptr->l);
cout<<ptr->info<<" ";
inorder(ptr->r);
}
}
void BST::postorder(nod *ptr)//postorder
traversal
{
if (r == NULL)
{
cout<<"Tree is empty"<<endl;
return;
}
if (ptr != NULL)
{
postorder(ptr->l);
postorder(ptr->r);
cout<<ptr->info<<" ";
}
}
void BST::show(nod *ptr, int
level)//print the tree
{
int i;
if (ptr != NULL)
{
show(ptr->r, level+1);
cout<<endl;
if (ptr == r)
cout<<"Root->: ";
else
{
for (i = 0;i < level;i++)
cout<<" ";
}
cout<<ptr->info;
show(ptr->l, level+1);
}
}
int main()
{
int c, n,item;
BST bst;
nod *t;
while (1)
{
cout<<"1.Insert Element "<<endl;
cout<<"2.Delete Element "<<endl;
cout<<"3.Search Element"<<endl;
cout<<"4.Inorder Traversal"<<endl;
cout<<"5.Preorder Traversal"<<endl;
cout<<"6.Postorder Traversal"<<endl;
cout<<"7.Display the tree"<<endl;
```

```
cout<<"8.Quit"<<endl;
cout<<"Enter your choice : ";
cin>>c;
switch(c)
{
case 1:
t = new nod;
cout<<"Enter the number to be inserted :
";
cin>>t->info;
bst.insert(r, t);
break;
case 2:
if (r == NULL)
{
cout<<"Tree is empty, nothing to
delete"<<endl;
continue;
}
cout<<"Enter the number to be deleted :
";
cin>>n;
bst.del(n);
break;
case 3:
cout<<"Search:"<<endl;
cin>>item;
bst.search(r,item);
break;
case 4:
cout<<"Inorder Traversal of BST:"<<endl;
bst.inorder(r);
cout<<endl;
break;
case 5:
cout<<"Preorder Traversal of BST:"<<endl;
bst.preorder(r);
cout<<endl;
break;
case 6:
cout<<"Postorder Traversal of
BST:"<<endl;
bst.postorder(r);
cout<<endl;
break;
case 7:
cout<<"Display BST:"<<endl;
bst.show(r,1);
cout<<endl;
break;
case 8:
exit(1);
default:
cout<<"Wrong choice"<<endl;
}
}
}
```