# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 20 |
| Roll No | 1333 |
| Unit | 6 |
| Program | Binary Search Trees |

**Source Code:**

#include<iostream> #include<conio.h> using namespace std;

/\* Node Template \*/ class BSTNode

{

public:

int data; BSTNode \*right; BSTNode \*left;

};

/\* 2. BST Template \*/ class BST

{

BSTNode \*root; int cnt;

public:

BST()

{

root = NULL; cnt = 0;

}

void Insert(int x); void Display();

void Inorder(BSTNode \*t); void Preorder(BSTNode \*t); void Postorder(BSTNode \*t); void Count();

# MET Institute of Computer Science

void Search(int x); void FindMax(); void FindMin();

};

/\* 3. functions \*/

//Insert

void BST :: Insert(int x)

{

//make a new node

BSTNode \*t = new BSTNode(); t->data = x;

t->left = NULL;

t->right = NULL;

//First node in the BST if(root == NULL)

{

root = t; cnt++; return;

}

//Traverse

BSTNode \*tmp = root; BSTNode \*prev = NULL; while(tmp!=NULL)

{

prev = tmp;

if(x > tmp->data)

{

tmp = tmp->right;

}

else if(x < tmp->data)

{

tmp = tmp->left;

}

else



# MET Institute of Computer Science

{

cout << "Duplicate value!"; getch();

return;

}

}

//Attach t to prev if(x < prev->data)

{

prev->left = t;

}

else

{

prev->right = t;

}

cnt++;

}

//Display

void BST :: Display()

{

if(root == NULL)

{

cout << "Empty Tree!"; return;

}

cout << "Inorder Traversal: "; Inorder(root);

cout << endl;

cout << "Preorder Traversal: "; Preorder(root);

cout << endl;

cout << "Postorder Traversal: "; Postorder(root);

}

//Inorder

# MET Institute of Computer Science

void BST :: Inorder(BSTNode \*t)

{

if(t)

{

Inorder(t->left);

cout << t->data << " "; Inorder(t->right);

}

}

//Preorder

void BST :: Preorder(BSTNode \*t)

{

if(t)

{

cout << t->data << " "; Preorder(t->left); Preorder(t->right);

}

}

//Postorder

void BST :: Postorder(BSTNode \*t)

{

if(t)

{

Postorder(t->left); Postorder(t->right); cout << t->data << " ";

}

}

//count

void BST :: Count()

{

if(root == NULL)

{

cout << "Empty Tree!";

# MET Institute of Computer Science

return;

}

else

{

cout << "Tree contains: " << cnt << " nodes";

}

}

//Search

void BST :: Search(int x)

{

if(root == NULL)

{

cout << "Empty Tree!"; return;

}

BSTNode \*tmp = root; int flag = 0;

while(tmp != NULL)

{

if(x < tmp->data)

{

tmp = tmp->left;

}

else if(x > tmp->data)

{

tmp = tmp->right;

}

else

{

flag = 1; break;

}

}//end of while if(flag == 0)

{

# MET Institute of Computer Science

cout << x << " is not found!";

}

else

{

cout << x << " is found!";

}

}

//findMax

void BST :: FindMax()

{

if(root == NULL)

{

cout << "Empty Tree!"; return;

}

BSTNode \*tmp = root; while(tmp->right != NULL)

{

tmp = tmp->right;

}

cout << "Maximum value: " << tmp->data;

}

//findMin

void BST :: FindMin()

{

if(root == NULL)

{

cout << "Empty Tree!"; return;

}

BSTNode \*tmp = root; while(tmp->left != NULL)

{

tmp = tmp->left;

# MET Institute of Computer Science

}

cout << "Minimum value: " << tmp->data;

}

/\* 4. Menu \*/ int main()

{

int num,ch; BST b;

while(1)

{

system("cls");

cout << "\*\*\*Binary Search Trees\*\*\*\n\n"; cout << "1. Insert" << endl;

cout << "2. Display the BST" << endl;

cout << "3. Count the nodes in the BST" << endl; cout << "4. Search for the element" << endl;

cout << "5. Find the maximum value" << endl; cout << "6. Find the minimum value" << endl; cout << "7. Exit" << endl;

cout << "\nEnter your choice: "; cin >> ch;

switch(ch)

{

case 1:

cout << "Enter the element: "; cin >> num;

b.Insert(num); getch(); break;

case 2: b.Display();

getch(); break; case 3: b.Count();

# MET Institute of Computer Science

getch(); break; case 4:

cout << "Enter the element: "; cin >> num;

b.Search(num); getch();

break; case 5:

b.FindMax();

getch(); break; case 6:

b.FindMin();

getch(); break; case 7:

exit(1); default:

cout << "Incorrect Option!"; getch();

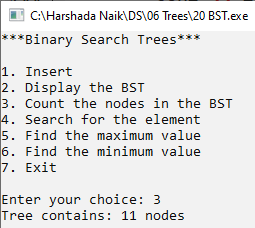
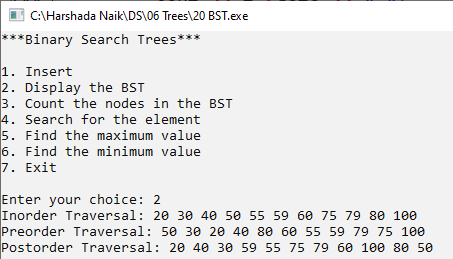
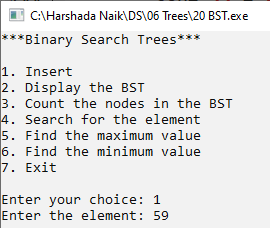
break;

}//end of switch

}//end of while

}//end of main

**Output: Insert:**

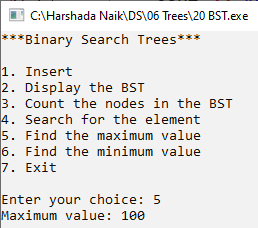
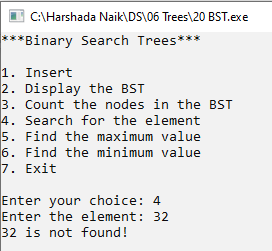
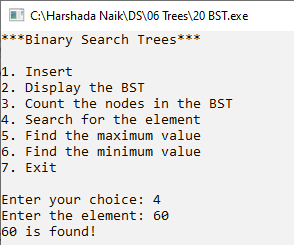


# MET Institute of Computer Science

**Display the BST**

**Count the nodes in trees:**

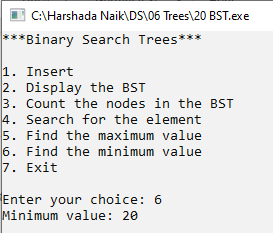
**When search is successful**



# MET Institute of Computer Science

**When search is unsuccessful**

**Find max value:**



# MET Institute of Computer Science

**Find Min value:**

# MET Institute of Computer Science

|  |  |
| --- | --- |
| Program No | 21 |
| Roll No | 1333 |
| Unit | 6 |
| Program | Max Heap |

#include<iostream> #include<conio.h> #define MAX 20 using namespace std;

/\* 1. Node Template \*/

/\* 2. Heap Template \*/ class Heap

{

int a[MAX]; int n; public:

Heap()

{

for(int i = 0; i<MAX; i++ )

{ a[i]=0;

}

n = 0;

}

void BuildHeap(); void Insert(int x); void DeleteHeap(); void Display();

void reheapDown(int pos); void reheapUp(int x);

};

/\* 3. Function \*/

void Heap :: BuildHeap()

{

int i;

cout << "Enter the size of the array: ";

# MET Institute of Computer Science

cin >> n; // n is declared in Heap Class

cout << "Enter " << n << " values in the Heap" << endl; for(i = 0; i < n; i++)

{

cin >> a[i];

}

for(i =(n/2)-1; i >=0;i--)

{

reheapDown(i);

}

}

void Heap :: Insert(int x)

{

a[n] = x; n++;

reheapUp(n-1);

}

void Heap :: reheapUp(int x)

{

int tmp = a[x];

while(a[(x-1)/2] <= tmp && x>0)

{

a[x] = a[(x-1)/2]; x = (x-1)/2;

}

a[x] = tmp;

}

void Heap :: reheapDown(int pos)

{

int tmp = a[pos]; int i;

while(pos <= (n/2)-1 ) // Comparing with last non-leaf position

{

i= (2 \* pos) + 1; // Left Child

if(a[i+1] > a[i]) // Compare left and right child

{

# MET Institute of Computer Science

i++;

}

if(tmp > a[i])

{

break;

}

a[pos] = a[i]; pos = i;

}

a[pos] = tmp;

}

void Heap :: DeleteHeap()

{

int x = a[0]; //copy root element to x

a[0] = a[n-1]; // copy last element to root

a[n-1] = 0; // Overwrite tle last element with 0 n--; // Decrement size of the heap reheapDown(0); // ReheapDown from root cout << "Element deleted is: " << x;

}

void Heap :: Display()

{

int i; for(i=0;i<n;i++)

{

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

}

}

/\* 4. Main \*/ int main()

{

int ch, num; Heap h; while(1)

{

system("cls");

# MET Institute of Computer Science

cout << "\*\*\* Max-Heap \*\*\*\n\n"; cout << "1. Build a heap"<< endl;

cout << "2. Insert an element" << endl; cout << "3. Delete" << endl;

cout << "4. Display" << endl; cout << "5. Exit" << endl; cout << "Enter your choice: "; cin >> ch;

switch(ch)

{

case 1: h.BuildHeap(); getch(); break;

case 2:

cout << "Enter an Element: "; cin >> num;

h.Insert(num); getch(); break;

case 3: h.DeleteHeap(); getch();

break; case 4:

h.Display();

getch(); break; case 5:

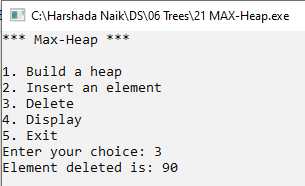
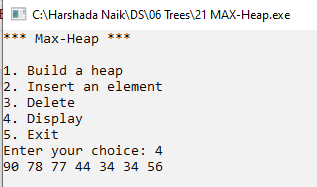
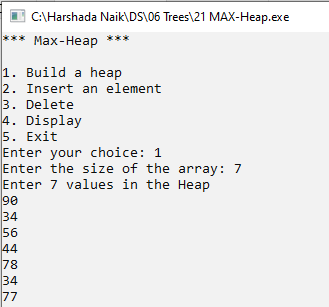
exit(1); default:

cout << "Incorect Choice!" << endl; getch();

break;

}

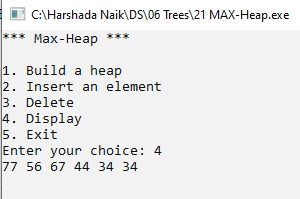
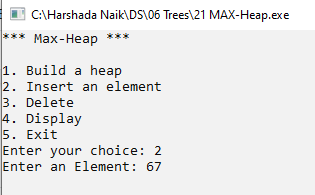
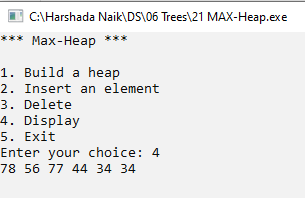
}



# MET Institute of Computer Science

}

**Output:**



# MET Institute of Computer Science