**Experiment No. 6c**

**Title :** Implementation of Search operation in Binary Search Tree

**Problem Statement :** Implementing search operation in binary search tree

Insert()

Search()

**Algorithm :**

**S1 :** Start

**S2 :** Create a struct variable as node which has data and left, right pointers.

**S3 :** initialize data to 0 and pointers to NULL.

**S4 :** In insert function if there is no value in tree add the node by constructor else check if value is greater or lesser than the existing node and decide if node is right or left.

**S5 :** In search function initialize a depth variable with value 0 and if temp node is not NULL keep incrementing and if the value searched is found print position else check if value is greater than nod or less than node and move accordingly.

**S6 :** Stop

**Code :**

#include<iostream>

using namespace std;

struct node {

int d;

node \*left;

node \*right;

};

node\* CreateNode(int d) {

node \*newnode = new node;

newnode->d = d;

newnode->left = NULL;

newnode->right = NULL;

return newnode;

}

node\* InsertIntoTree(node\* root, int d) {

node \*temp = CreateNode(d);

node \*t = new node;

t = root;

if(root == NULL)

root = temp;

else {

while(t != NULL) {

if(t->d < d) {

if(t->right == NULL) {

t->right = temp;

break;

}

t = t->right;

} else if(t->d > d) {

if(t->left == NULL) {

t->left = temp;

break;

}

t = t->left;

}

}

}

return root;

}

void Search(node \*root, int d) {

int depth = 0;

node \*temp = new node;

temp = root;

while(temp != NULL) {

depth++;

if(temp->d == d) {

cout<<"\nitem found at depth: "<<depth;

return;

} else if(temp->d > d)

temp = temp->left;

else

temp = temp->right;

}

cout<<"\n item not found";

return;

}

int main() {

char ch;

int n, i, a[10] = {93, 53, 45, 2, 7, 67, 32, 26, 71, 76};

node \*root = new node;

root = NULL;

for (i = 0; i < 10; i++)

root = InsertIntoTree(root, a[i]);

up:

cout<<"\nEnter the Element to be searched: ";

cin>>n;

Search(root, n);

cout<<"\n\n\tDo you want to search more...enter choice(y/n)?";

cin>>ch;

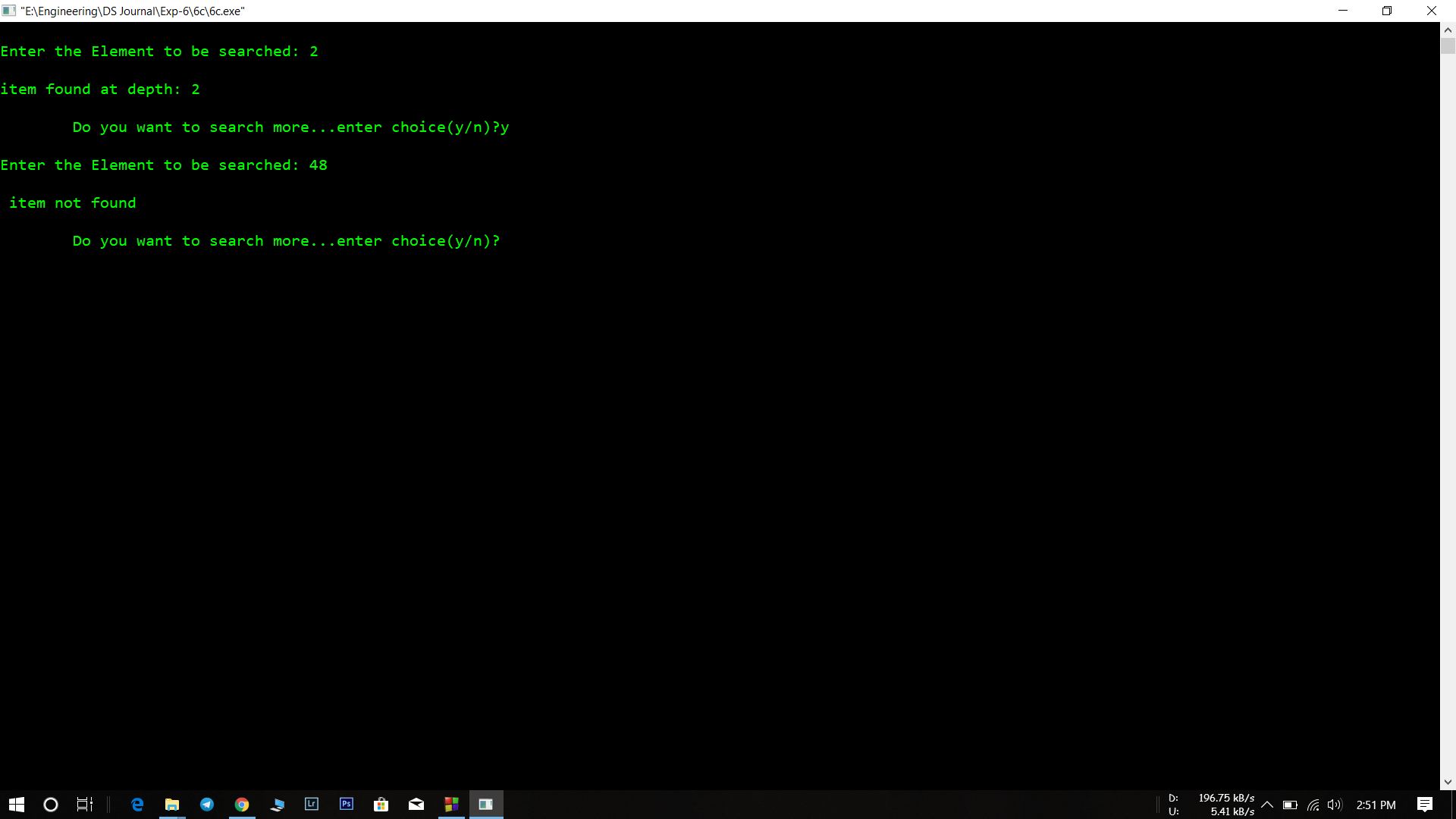
if(ch == 'y' || ch == 'Y')

goto up;

return 0;

}

**Output :**



**Analysis :**

* The searching in binary tree is very fast as compared to one or two dimensional array.
* The time complexity increases as the nodes increase since to add a node we have to traverse until the leaf node on any one side.