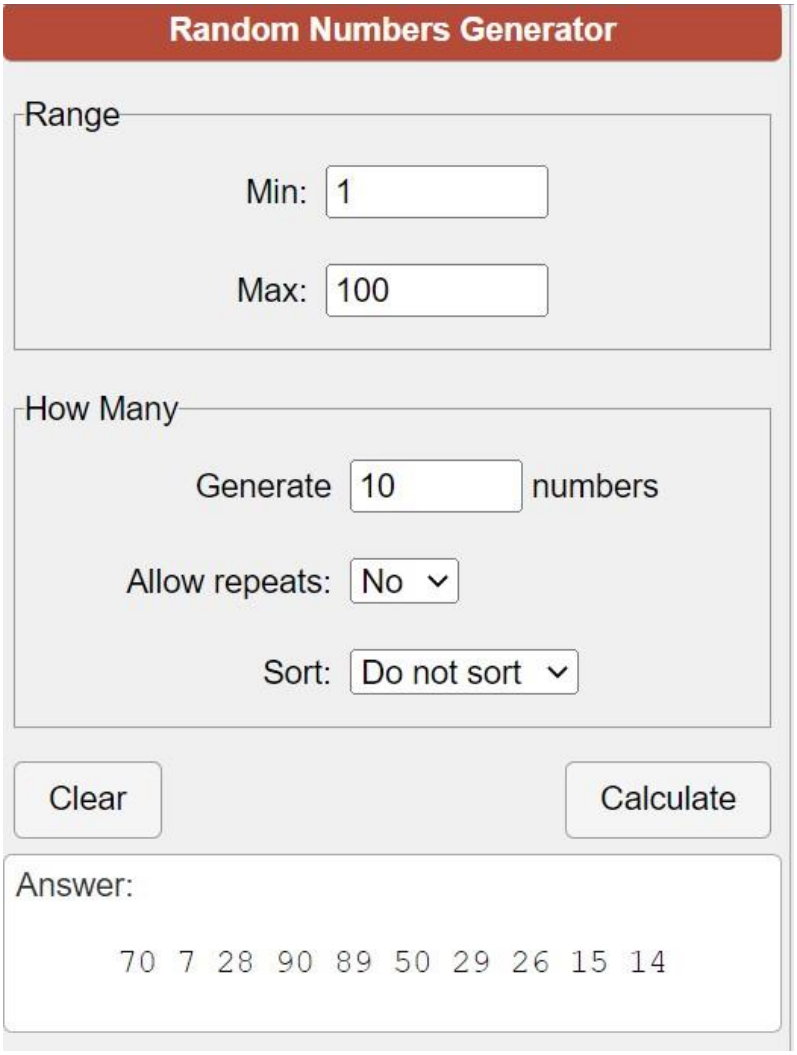
Code Description

So I created a Data Structure That is Called BST. basically BST is a class Which has its nodes information about them is in TreeNodes.h where you can see that every node knows the address of its left and right nodes and all of them have their independent data as well of course. Node of the BST is also another class which is not visible for User but actually he /she/it creates it whenever they'll do some manipulations with The Class objects . For the first(second) testing of the manipulations I am testing the overloaded assignment operator , copy constructor , Dynamic memory allocation and test whether I have created them correctly in the implementations whether there were errors or not .For that I am creating one say so main object of BST class and then delete some of them oh yeah in the beginning I have checked whether the bst was empty or not since with helping of default constr there I had an empty object whenever I have inserted 10 randomly generated number in my tree I deleted some of them and denote that the tree is not empty anymore After that this say so main bst object was used to manipulate to upward stated things , first of all I came up with overloaded assignment operator , besides to assignment there is overloaded ostream expression operator overloaded as well with helping of what I simply gain the possibility to invoke display function with just cout<<classobj; writing; for checking logical errors I have also written try catch staff as well if after default constr overloaded won't work then I will stop there and say that okay this is an error the programme doesn't working sufficiently but fortunately during this testing I had not have such an exception.User has an ability to provide the keyword and the programme by itself determine whether it is in BST or not besides , There is also separate method which checking the data existence in BST and also determines the depth of it if it is found obviously.For basicmethod I have created there you can see both recursive and iterative methods in the Test2() I have most of the time iterative ones in the Test1() recursive ones is dominant.

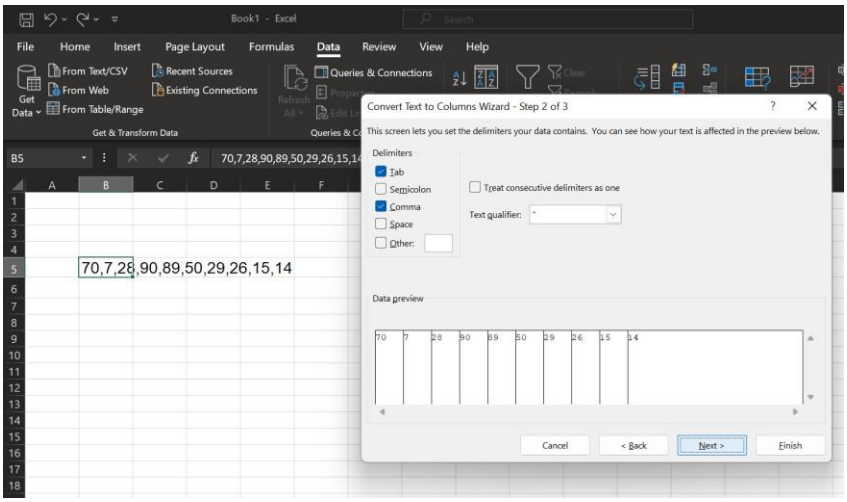
Contents

For Time Saving…

From the range of 1 to 100 I used online source in order to randlomy generate numbers



Then i used Ms excel After passing those numbers in excel cells with helping of Text to columns command I split those values in separate cells



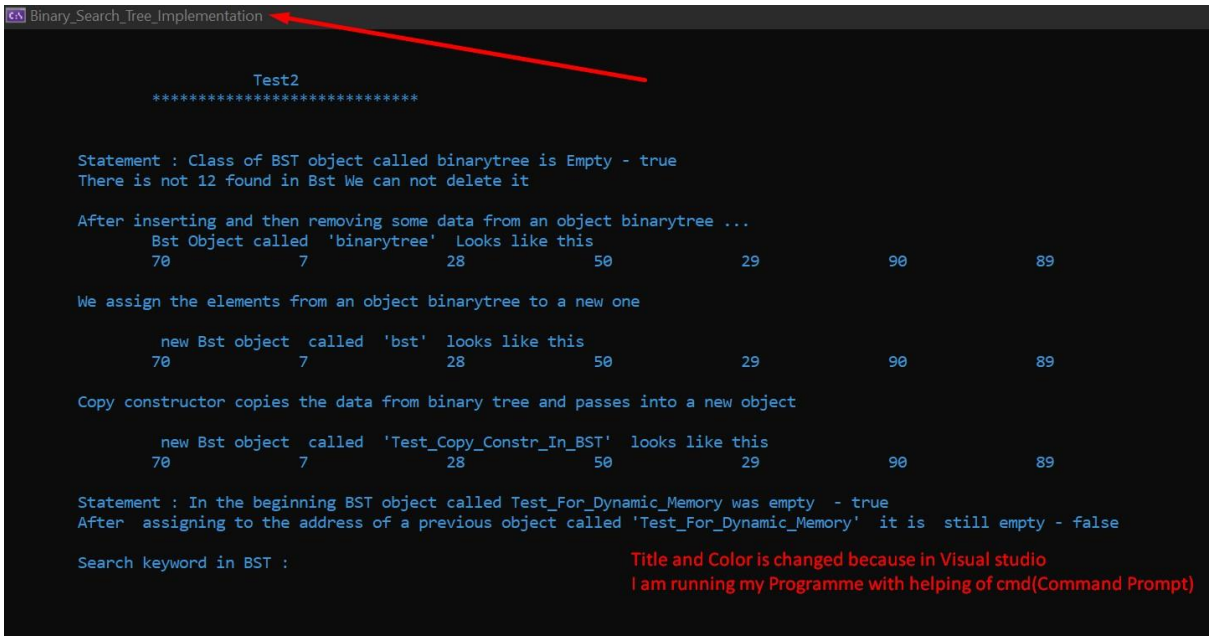
Ah yeah I delimite those values by commas Then Transpose the data

Since - transposed data is copied into the clipboard I drag and drop it and then copied I used keyboard shortcuts Shift+Alt+Down Arrow and pass the data as a parameters of the insert_an_item() method so simply wrote the second Testing of the programe

```
64    void Test1() { ... }
127
128   void Test2() {
129       BST binarytree,bst;
130       std::cout << "\n\n\t\t\t\Test2" << std::endl;
131       std::cout<<"\tClass of BST object called binarytree is
132       binarytree.Insert_an_item(70);
133       binarytree.Insert_an_item(7);
134       binarytree.Insert_an_item(28);
135       binarytree.Insert_an_item(90);
136       binarytree.Insert_an_item(89);
137       binarytree.Insert_an_item(50);
138       binarytree.Insert_an_item(29);
139       binarytree.Insert_an_item(26);
140       binarytree.Insert_an_item(15);
141       binarytree.Insert_an_item(14);
142
143       binarytree.Display_Tree(std::cout);
144
```

Several Run-time Screenshots

We're Starting with the second Testing … (system (color … title…))



Error Checking with providing Searching Keywords

```
        Search keyword in BST : LET's

Error Occured  -  Invalid Input Provide again : search

Error Occured  -  Invalid Input Provide again : keyword

Error Occured  -  Invalid Input Provide again : oh come on what's happening

Error Occured  -  Invalid Input Provide again : oh I just read the information

Error Occured  -  Invalid Input Provide again : okay

Error Occured  -  Invalid Input Provide again : 12


        Statement that  12 is  in Bst is false
```

Input the Data in an empty BST structure , the data validation

```
       You have to Provide  item to insert into an empty new Object
 Please be informed that you have to Provide the items  till the it  won't be 0  so you can not insert 0

       Provide The item : 12

       Provide The item : 12

Duplicate values can not be passed in BST , Value is Already There

       Provide The item : 23

       Provide The item : 90

       Provide The item : 89

       Provide The item : 67

       Provide The item : 12

Duplicate values can not be passed in BST , Value is Already There

       Provide The item : 78

       Provide The item : 0

       After Insertion  emptynewObject 's data is          12          23          90          89          67          78
       Delete two elements from an object

       Pass the number here to delete :
```

Data Validation in BST They can not be duplicates

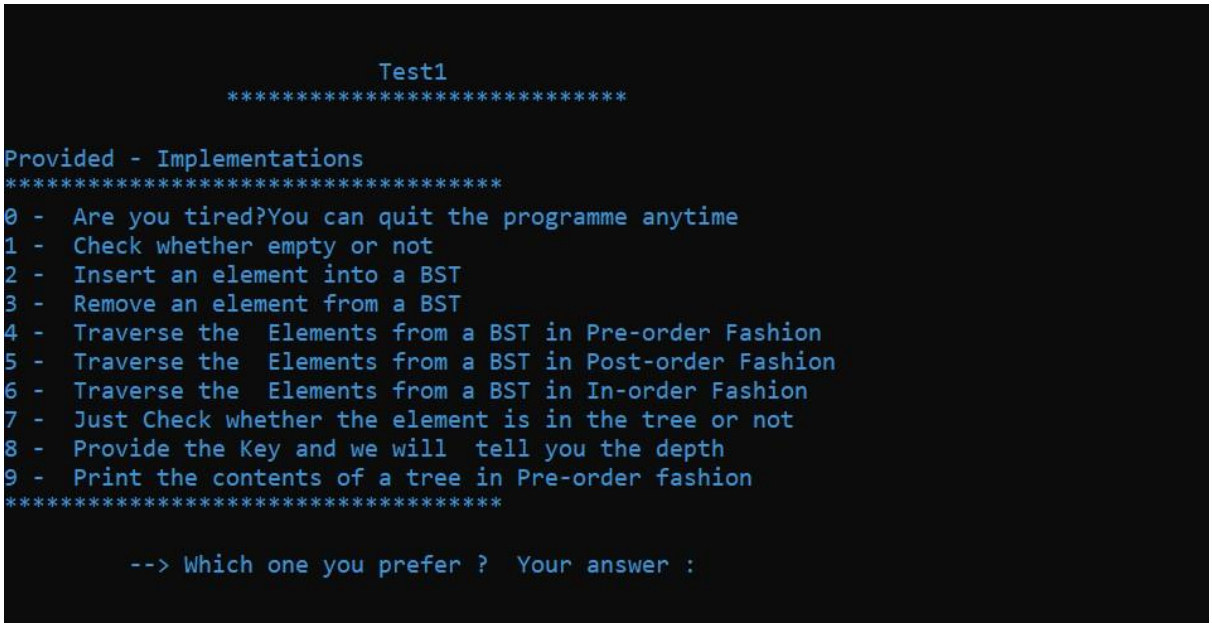
Data Removing Validation

```
       After Insertion  emptynewObject 's data is          12          23          90          89          67          78
       Delete two elements from an object

       Pass the number here to delete : 5

Keyvalue you are providing to delete is not in BST at all we can not delete

       Pass the number here to delete : 12

       After Deletion :            12          23          90          89          67          78
************************** End of The  Testing 2  **************************
```

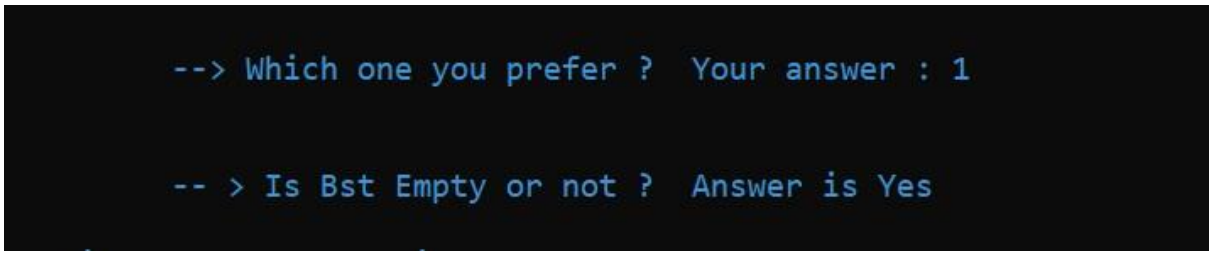
Our Data

Provided Numbers is not in BST so we can not delete it Validation

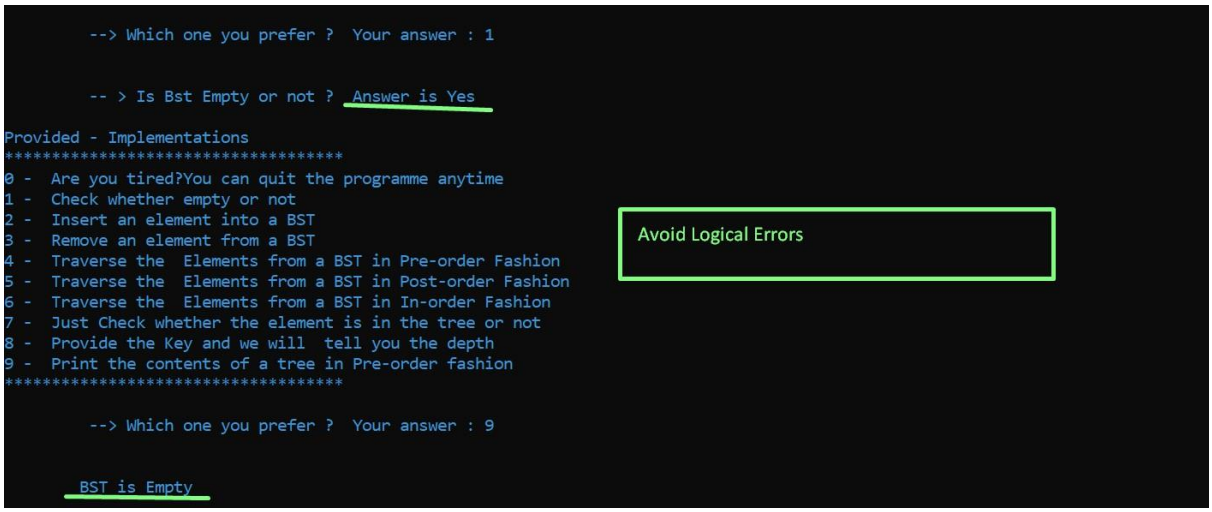
Test 1 is more user-defined It has the General menu



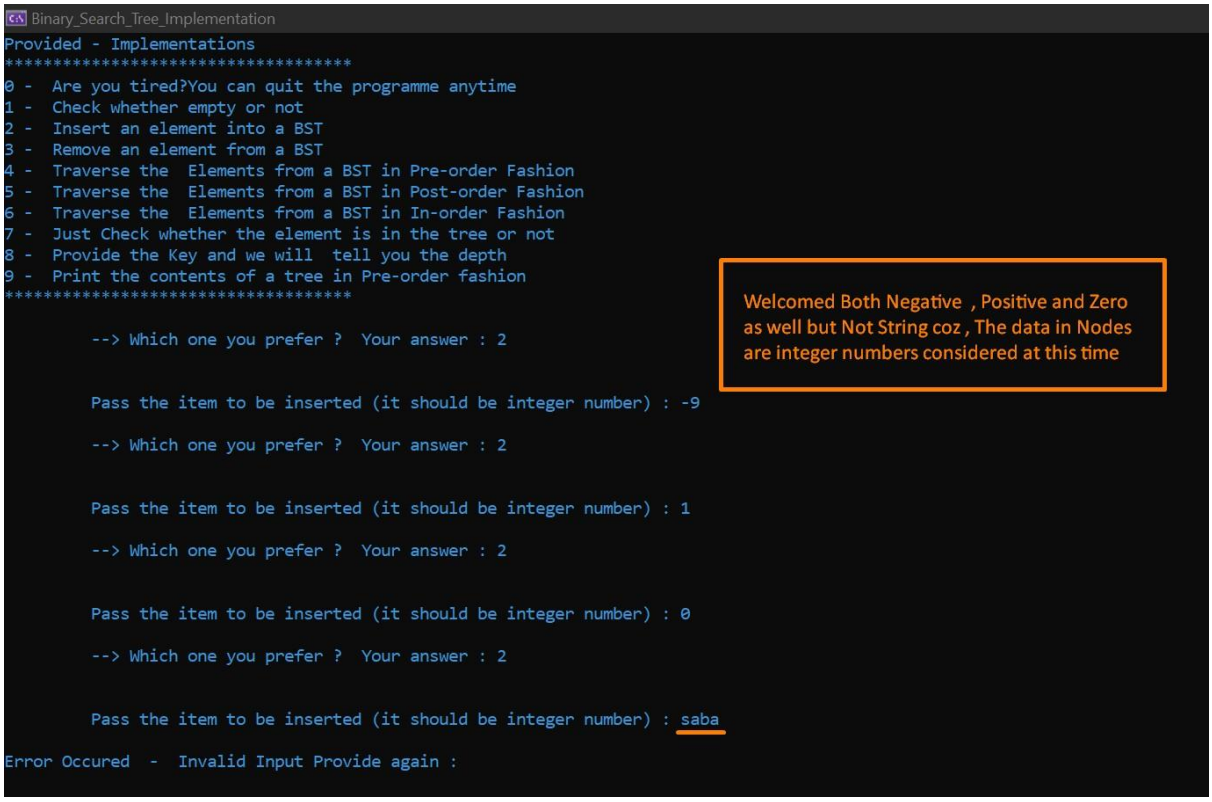
Case User chose 1 : Checking if BST is an empty or not



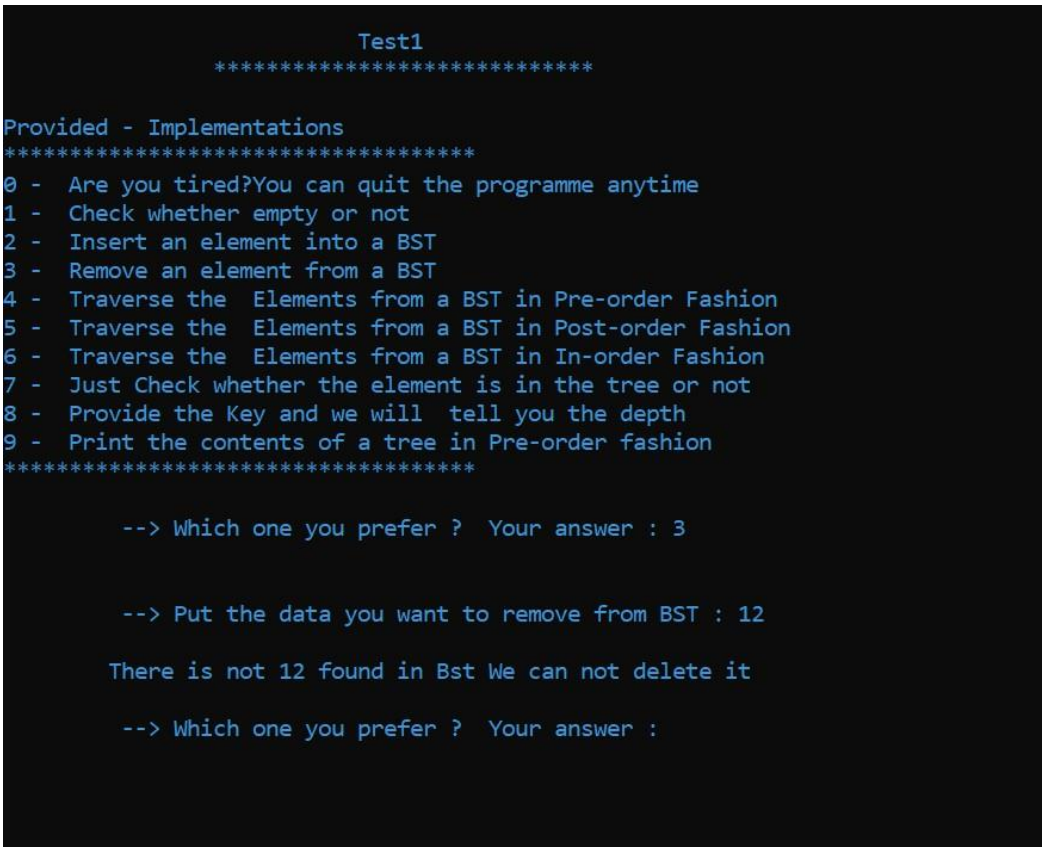
Let's Try to traverse such BST content



Case 2 : Push some elements into BST



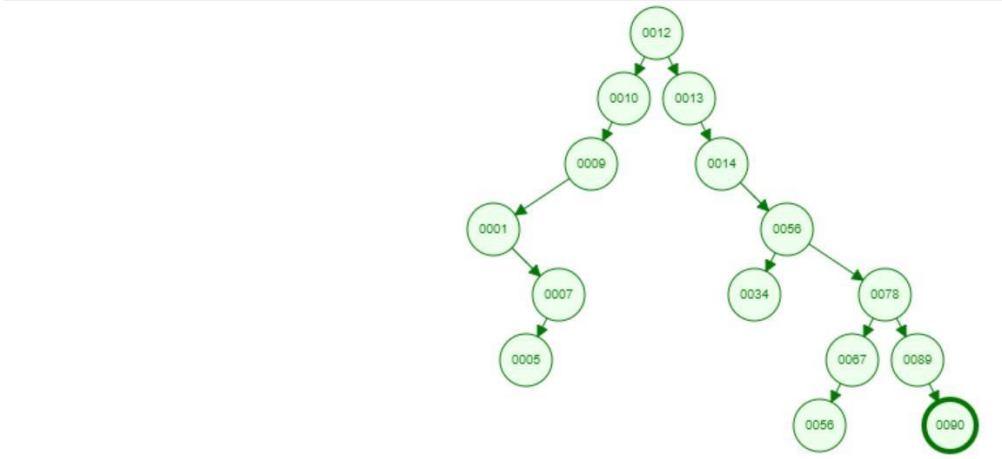
Case 3: When the item is not found into the BST of course we can not delete it



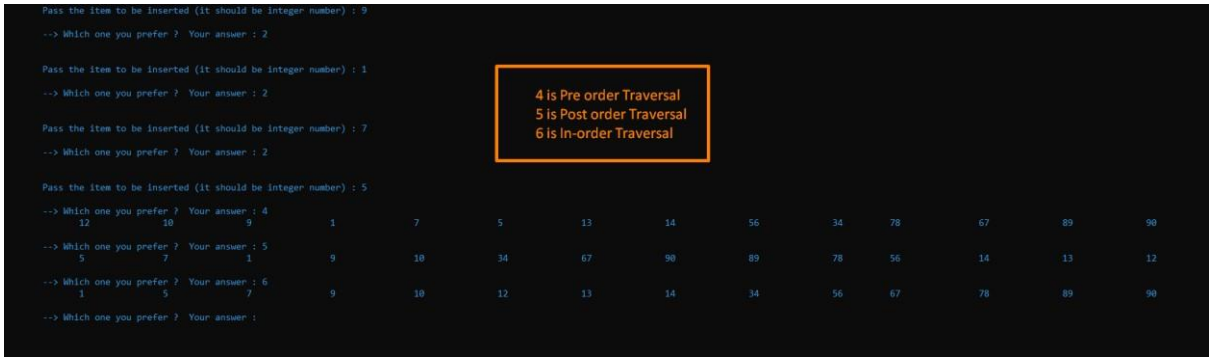
However I pushed it and then remove So removing comes successfully there since there is not any additional comment

```
                        Test1
                ****************************

Provided - Implementations
**********************************
0 -  Are you tired?You can quit the programme anytime
1 -  Check whether empty or not
2 -  Insert an element into a BST
3 -  Remove an element from a BST
4 -  Traverse the  Elements from a BST in Pre-order Fashion
5 -  Traverse the  Elements from a BST in Post-order Fashion
6 -  Traverse the  Elements from a BST in In-order Fashion
7 -  Just Check whether the element is in the tree or not
8 -  Provide the Key and we will  tell you the depth
9 -  Print the contents of a tree in Pre-order fashion
**********************************

        --> Which one you prefer ?  Your answer : 3


        --> Put the data you want to remove from BST : 12

       There is not 12 found in Bst We can not delete it

        --> Which one you prefer ?  Your answer : 2


        Pass the item to be inserted (it should be integer number) : 12

        --> Which one you prefer ?  Your answer : 3


        --> Put the data you want to remove from BST : 12

        --> Which one you prefer ?  Your answer :
```

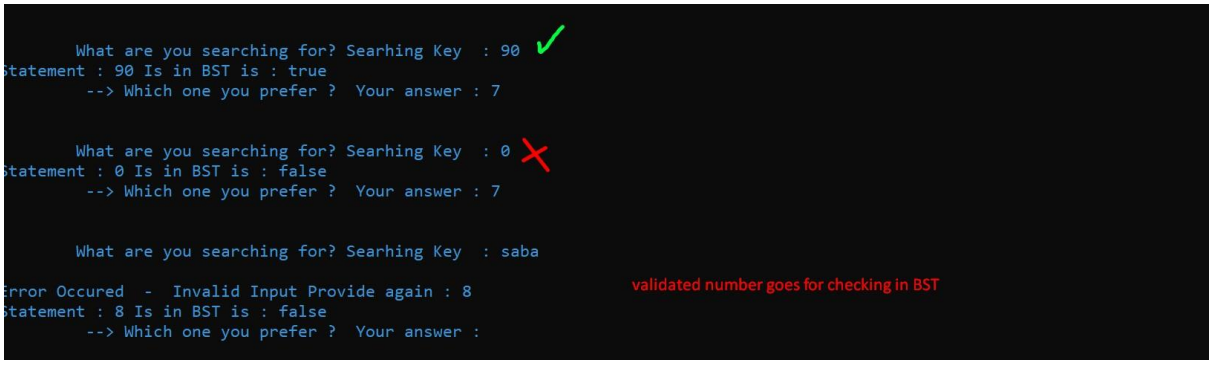
Case 4 (9) - 5 - 6 : Inserted Data visualized (source)



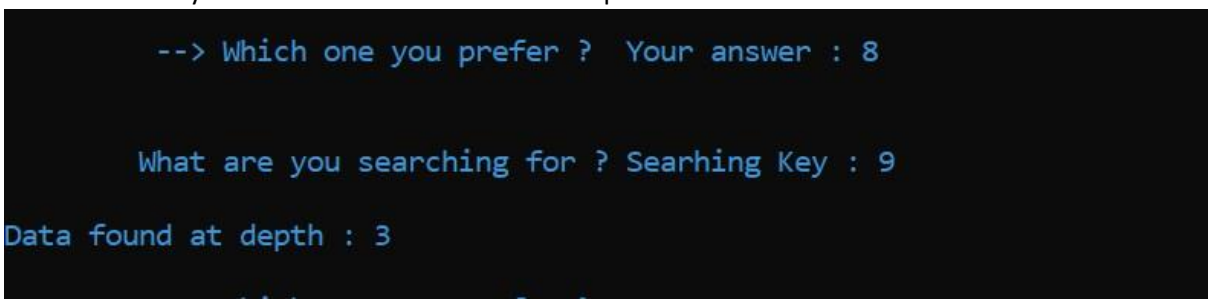
Traversing The elements Inserted in Three Different Types

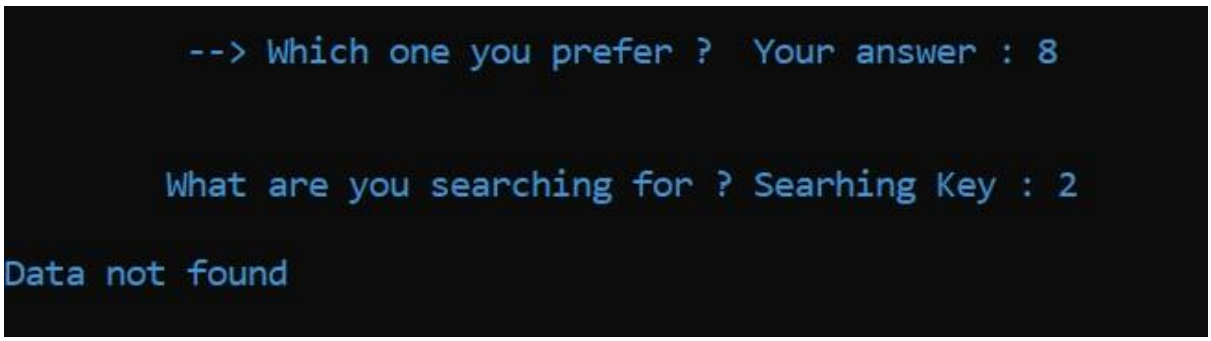


Case 7 : Searching Item In BST

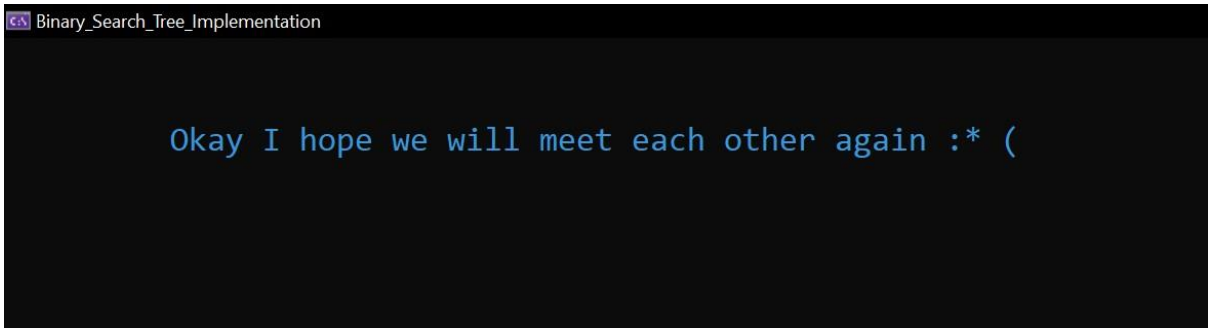


Case 8 : Not only search but also determine the depth as well





Case 0 : If the user is so tired He/she/it can Quit The programme with typing 0



Code In C++

TreeNodes.h

```
#ifndef Assign4_Bsth_Treenode_H #define
Assign4_Bsth_Treenode_H
typedef int Datatype; // do some implementations with integers then another data
types and soon class TreeNode {
	friend class BST;//it should be friend according to our instruction
public:
	Datatype data = 0;
	TreeNode* left = nullptr, * right = nullptr; // by default data is
initialized  to zero and the pointers to right	//and left are
nullpointers as well
	TreeNode() = default;  //Default constructor
 TreeNode(Datatype Data, TreeNode* Left = nullptr, TreeNode* Right = nullptr) :
  data(Data), left(Left), right(Right) {} //Explicite  value constructor
	Datatype GetItem()const { return data; } //data accessor
};  #endif
```

BST.h

```
#ifndef Assign4_Binarystree_BST_H
#define Assign4_Binarystree_BST_H


#include <iostream>
#include "Treenode.h"


class BST {

	TreeNode* root = nullptr;

public:

	BST() = default;//default  constr

	bool IsEmpty();

	BST(const BST&); //Copy constructor


	~BST(); //Destructor
```

```
	void In_Order_Traversal(); //Displays the elements of BST  inorder fashion

	void  Pre_Order_Traversal();    // Displays the elements of BST preorder
fashion

	void  Post_Order_Traversal();  // Displays the elements of BST postorder
fashion

	void Insert_an_item (Datatype item);  //Inserts item in BST

	void Delete(Datatype fet_deleted);   //Deletes given item form BST if it is
in so

	bool Search(Datatype data);  //Searchs item in BST and retuns true if it is
found else returns false

	//searches with boolean if it is true finds it depth and then traverse

	BST& operator =(const BST&);  //Overloading assignment operator

	void Display_Tree(std::ostream&);

	void Search_find_Depth(int Key);

	bool iterative_Search(Datatype);

	void Insert_an_item_iteratively(Datatype);

	void Delete_an_item_iteratively(Datatype);

private:
	//  Helper functionalities in the class private section
	void HelpInCopy(TreeNode*);  //Copy Constructor Helper
	//Destructor Helper
	void HelpInDelete(TreeNode*&);
	//Inoder traversial helper
	void In_Order_Traversal(TreeNode*);
	//Postorder traversial helper
	void  Post_Order_Traversal(TreeNode*);
	//Preorder traversial helper
	void  Pre_Order_Traversal(TreeNode*);
	//Insertion helper
	void Insert_an_item (TreeNode*&, Datatype);
	//Deletiong helper
	void Delete(TreeNode*&, Datatype);
	//Search helper
	bool Search(TreeNode*, Datatype);
	//Search iteratively
}; #endif
```

BST.cpp

```
#include "BST.h"


BST::BST(const BST& B2) {
	HelpInCopy(B2.root);
}//Copty constructor


void BST::HelpInCopy(TreeNode* Root) {
```

```
//Iterating inorder to add copy item  with helper functions
// because of we need to output things  in pre-order traversal methodic
(according to the instruction ) I am copying respectively  from original to
another one
	if (Root != nullptr) {
		Insert_an_item(Root->data);
		HelpInCopy(Root->right);
		HelpInCopy(Root->left);
	}
}  //Copy Constructor Helper

BST::~BST() {
	HelpInDelete(root);
}    //Destructor

void BST::HelpInDelete(TreeNode*& Root) {
	if (Root != nullptr) {
		HelpInDelete(Root->left);
		HelpInDelete(Root->right);
		delete Root;
		Root = nullptr;
	}
} //Destructor Helper

void BST::In_Order_Traversal() {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	In_Order_Traversal(root);
	std::cout << std::endl;
}// inorder traversal method


void BST:: Pre_Order_Traversal() {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	 Pre_Order_Traversal(root);
	std::cout << std::endl;
}//pre-order traversal method

void BST:: Post_Order_Traversal() {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	 Post_Order_Traversal(root);
	std::cout << std::endl;
}

void BST::In_Order_Traversal(TreeNode* Root) {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	if (Root != nullptr) {
		In_Order_Traversal(Root->left);
		std::cout <<"\t\t" << Root->data << " ";
		In_Order_Traversal(Root->right);
	}
}//Inorder traversal helper
```

```
void BST:: Post_Order_Traversal(TreeNode* Root) {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	if (Root != nullptr) {
		 Post_Order_Traversal(Root->left);
		 Post_Order_Traversal(Root->right);
		std::cout <<"\t\t" << Root->data << " ";
	}
}//Postorder traversal helper

void BST:: Pre_Order_Traversal(TreeNode* Root) {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	if (Root != nullptr) {
		std::cout <<"\t\t" << Root->data << "  ";
		 Pre_Order_Traversal(Root->left);
		 Pre_Order_Traversal(Root->right);
	}
}//Preorder traversal helper

void BST::Insert_an_item (Datatype data) {
	Insert_an_item (root, data);
}//Insert item in BST

void BST::Insert_an_item (TreeNode*& Root, Datatype data) {
	if (Root == nullptr) {
		//At this point we have found the place to insert the node
		Root = new TreeNode(data);
	}
	else if (Root->data == data) {
		std::cout << "\n\aDuplicate values can not be passed in BST , Value
is Already There" << std::endl;
	}
	else if (Root->data < data) {
		Insert_an_item (Root->right, data);
	}
	else if (Root->data > data) {
		Insert_an_item (Root->left, data);
	}
}//Insertion helper

void BST::Delete(Datatype data) {
	Delete(root, data);
}//Deletes item from BST

void BST::Delete(TreeNode*& Root, Datatype data) {
	if (Root == nullptr)
		std::cout << "\a\n\tThere is not "<<data<<" found in Bst We can not
delete it" << std::endl;
	else if (Root->data > data)
		Delete(Root->left, data);
	else if (Root->data < data)
		Delete(Root->right, data);
	else {
		//At this point we have found the node that should be deleted
		if (Root->left == nullptr) {
			TreeNode* temp = Root;
```

```
                        Root = Root->right;
                        delete temp;
                }
                else if (Root->right == nullptr) {
                        TreeNode* temp = Root;
                        Root = Root->left;
                        delete temp;
                }
                else if (Root->left != nullptr && Root->right && nullptr) {

                        //Lets find max form left subtree
                        TreeNode* MaxFromLeftSubTree = Root->left;
                        while (MaxFromLeftSubTree->right != nullptr)
                                MaxFromLeftSubTree = MaxFromLeftSubTree->right;
                        //Exchange values and delete
                        Root->data = MaxFromLeftSubTree->data;
                        delete MaxFromLeftSubTree;
                        MaxFromLeftSubTree = nullptr;
                }
        }
}//Deletiong helper

bool BST::Search(Datatype data) {
        return Search(root, data);
}//Searchs item in BST and retuns boolean


bool BST::Search(TreeNode* Root, Datatype data) {
        if (Root == nullptr)
                return false;
        if (Root->data > data)
                Search(Root->left, data);
        else if (Root->data < data)
                Search(Root->right, data);
        else if (Root->data = data)
                return true;
}//recursively searches

bool BST::iterative_Search(Datatype keyword)
{
        while (root != NULL) {
                if (keyword > root->data)
                        root = root->right;
                else if (keyword < root->data)
                        root = root->left;
                else
                        return true;
        }
        return false;
}


void BST::Insert_an_item_iteratively(Datatype data) {
        TreeNode* temp = root;
        TreeNode* mom = 0;
        bool found = false;
        while (!found && temp != 0) {
                mom = temp;
                if (data < temp->data)
                        temp = temp->left;
                else if (data > temp->data)
                        temp = temp->right;
                else
```

```
                found = true;
        }
        if (!found) {
                temp = new TreeNode(data);
                if (mom == 0)
                        root = temp;
                else if (data < mom->data)
                        mom->left = temp;
                else
                        mom->right = temp;
        }
        else {
                std::cout << "\n\aDuplicate values can not be passed in BST , Value
is Already There" << std::endl;
        }
}

void BST::Delete_an_item_iteratively(Datatype keyvalue) {
        TreeNode* temp = root;
        TreeNode* mom = 0;
        bool found = false;
        while (!found && temp != 0) {
                mom = temp;
                if (keyvalue < temp->data)
                        temp = temp->left;
                else if (keyvalue > temp->data)
                        temp = temp->right;
                else
                        found = true;
        }
        if (!found)
        {
                std::cout << "\a\nKeyvalue you are providing to delete is not in BST
at all we can not delete \n";
                return;
        }
        if(temp->left != 0 && temp->right!= 0)
        {
                TreeNode* temporary = temp->right;
                mom = temp;
                while (temporary->left != NULL)
                {
                        mom = temporary;
                        temporary = temporary->left;
                }
                temp->data = temporary->data;
                temp = temporary;
        }
        TreeNode* Stree = temp->left;
        if (Stree == 0)
                Stree = temp->right;
        if (mom == 0)
                root = Stree;
        else if (mom->left == temp)
                mom->left = Stree;
        else
                mom->right = Stree;
        //delete temp;
}

//Overloading assignment operator

BST& BST::operator = (const BST& B2) {
```

```
	//First of all clearing up this tree
	HelpInDelete(root);
	HelpInCopy(B2.root);
	return *this;
}

void BST::Display_Tree(std::ostream&) {
	if (IsEmpty()) {
		std::cout << "\n\n\tBST is Empty" << std::endl;
		return;
	}
	Pre_Order_Traversal();
}

void BST::Search_find_Depth(int Key)
{
	int depth = 0;
	TreeNode* temp = new TreeNode;
	temp = root;
	while (temp != NULL)
	{
		depth++;
		if (temp->data == Key)
		{
			std::cout << "\nData found at depth : " << depth << std::endl;
			return;
		}
		else if (temp->data > Key)
			temp = temp->left;
		else
			temp = temp->right;
	}
	std::cout << "\nData not found " << std::endl;
	return;
}

bool BST::IsEmpty() {
	return root == NULL;
}
```

Driver.cpp

```
#include "Bst.h"
#include <vector>
#include <limits>
 void
Test1(); void
Test2();

std::ostream& operator<<(std::ostream& COUT, BST& tree) {
tree.Display_Tree(COUT);		return COUT;
}
std::string yes_or_not(bool statement ) {
return (statement) ? "Yes" : "No";
}
template <typename Datatype> Datatype validate_(Datatype Data) {
	while (std::cin.fail()) {
std::cin.clear();
			std::cin.ignore(1000000000, '\n');
std::cout << "\n\aError Occured  -  Invalid Input ";
			std::cout << "Provide again : "; std::cin >> Data;
			//validate_(Data);
```

```
		}
		return Data;
}
void ShowMenu() {
	std::cout << "\nProvided - Implementations "
	<< "\n************************************"
		<< "\n0 -  Are you tired?You can quit the programme anytime "
		<< "\n1 -  Check whether empty or not"
		<< "\n2 -  Insert an element into a BST"
		<< "\n3 -  Remove an element from a BST"
		<< "\n4 -  Traverse the  Elements from a BST in Pre-order Fashion"
		  << "\n5 -  Traverse the  Elements from a BST in Post-order Fashion"
		<< "\n6 -  Traverse the  Elements from a BST in In-order Fashion"
		<< "\n7 -  Just Check whether the element is in the tree or not"
		<< "\n8 -  Provide the Key and we will  tell you the depth "
		<< "\n9 -  Print the contents of a tree in Pre-order fashion "
		<< "\n************************************" << std::endl; }
int menu() {
	int your_answer, final_result;  static int temp = 0;
if (temp == 0)
		ShowMenu();
	++temp;
	std::cout << "\n\t --> Which one you prefer ?  Your answer : ";  std::cin
>> your_answer;
	final_result = validate_<int>(your_answer);
	if (final_result != 1 and
final_result != 2 and		final_result
!= 3 and		final_result != 4 and
	final_result != 5 and
final_result != 6 and		final_result
!= 7 and		final_result != 8 and
	final_result != 9 and
		final_result != 0)
	{
		std::cout << "\a\n\t --> Please provide the namber betweeen 0 to 9
" << std::endl;
		return  -1;
	}

	return final_result;
}  int
main() {
	system("title Binary_Search_Tree_Implementation");
system("color 3");
	Test1();
	//Test2();
}  void
Test1() {
	std::cout << "\n\n\t\t\t   Test1" << std::endl;
	std::cout << "\t\t*****************************" << std::endl;
	BST bst;
int result;
do {
		result = menu();

		if (result == 1) {
			std::cout << "\n\n\t -- > Is Bst Empty or not ?  Answer is "
<< yes_or_not(bst.IsEmpty()) << std::endl;
continue;
```

```
		}
		else if (result == 2) {
			Datatype item;
			std::cout << "\n\n\t Pass the item to be inserted (it should
be integer number) : "; std::cin >> item;
			item = validate_(item);
			bst.Insert_an_item(item);
			continue;
		}
		else if (result == 3) {
	Datatype deleted;
			 std::cout << "\n\n\t --> Put the data you want to remove from
BST : "; std::cin >> deleted;
			bst.Delete(deleted);
			continue;
		}
		else if (result == 4) {

			bst.Pre_Order_Traversal();
			continue;
		}
		else if (result == 5) {

			bst.Post_Order_Traversal();
			continue;
		}
		else if (result == 6) {
bst.In_Order_Traversal();
			continue;
		}
		else if (result == 7) {
	Datatype searhing_item;
			std::cout << "\n\n\tWhat are you searching for? Searhing Key
: "; std::cin >> searhing_item;
			searhing_item = validate_(searhing_item);
std::cout<<"Statement : "<< searhing_item <<" Is in BST is :
"<< std::boolalpha << bst.Search(searhing_item);
			continue;
		}
		else if (result == 8) {
			Datatype Key;
			std::cout <<"\n\n\tWhat are you searching for ? Searhing Key
: "; std::cin >> Key;
			Key = validate_(Key);
			bst.Search_find_Depth(Key);
			continue;
		}
		else if (result == 9) {
std::cout << bst;		continue;
		}
		else if (result == 0) {
system("cls");
   std::cout << "\n\n\t Okay I hope we will meet each other again :* ( " <<
std::endl;
			system("pause>0");
		break;
		}
	} while (true);
```

```
}  void
Test2() {
	system("cls");//clears everything that was before written on cmd
	//system("color 5");// gives a color the text in cmd
	BST binarytree, bst;
	std::cout << "\n\n\t\t\t   Test2" << std::endl;     std::cout <<
"\t\t*****************************" << std::endl;   std::cout <<
"\n\n\tStatement : Class of BST object called binarytree is
Empty - " << std::boolalpha << binarytree.IsEmpty();
binarytree.Insert_an_item(70);   binarytree.Insert_an_item(7);
binarytree.Insert_an_item(28);   binarytree.Insert_an_item(90);
binarytree.Insert_an_item(89);   binarytree.Insert_an_item(50);
binarytree.Insert_an_item(29); binarytree.Insert_an_item(26);
binarytree.Insert_an_item(15); binarytree.Insert_an_item(14);
	//after insertion of some elements let's delete  some of them

	binarytree.Delete(26);
binarytree.Delete(15);    binarytree.Delete(14);
binarytree.Delete(12);



	//then let's Traverse
	std::cout << "\n\tAfter inserting and then removing some data from an
object binarytree ... \n\t\tBst Object called  'binarytree'  Looks like this " <<
std::endl << binarytree;




	//test overloaded assignment operator
bst = binarytree;
	std::cout << "\n\tWe assign the elements from an object binarytree to a
new one \n\n\t\t new Bst object  called  'bst'  looks like this " << std::endl <<
bst;

	//test copy constr
	BST Test_Copy_Constr_In_BST = binarytree;
	std::cout << "\n\tCopy constructor copies the data from binary tree and
passes into a new object \n\n\t\t new Bst object  called
'Test_Copy_Constr_In_BST'  looks like this " << std::endl;
	Test_Copy_Constr_In_BST.Display_Tree(std::cout);




	//check for dynamic memory and then assignment
BST* Test_For_Dynamic_Memory = new BST;
	std::cout << "\n\tStatement : In the beginning BST object called
Test_For_Dynamic_Memory was empty  - " << std::boolalpha <<
Test_For_Dynamic_Memory->IsEmpty();
	Test_For_Dynamic_Memory = &binarytree;
	try {
		std::cout << "\n\tAfter  assigning to the address of a previous
object called 'Test_For_Dynamic_Memory'  it is  still empty - " << std::boolalpha
<< Test_For_Dynamic_Memory->IsEmpty();
if (Test_For_Dynamic_Memory->IsEmpty())
			throw std::logic_error("err");
	}
```

```
	catch (std::logic_error& err) {
		std::cout << "\n\tOverloaded asssignment operator is not working
Properly . . . " << std::endl;
	}
	Datatype Keyword;
	std::cout << "\n\n\tSearch keyword in BST : "; std::cin >> Keyword; Keyword
= validate_<Datatype>(Keyword);
	std::cout << std::endl << "\n\n\tStatement that  "<<Keyword<<" is  in Bst
is " <<std::boolalpha<<binarytree.iterative_Search(Keyword) << std::endl;
	Datatype Item = 9;
BST emptynewObject;
	std::cout << "\n\n\tYou have to Provide  item to insert into an empty new
Object  \n Please be informed that you have to Provide the items  till the it
won't be 0  so you can not insert 0 "<<std::endl;
 while (Item != 0) { std::cout << "\n\n\tProvide The item : "; std::cin
	>> Item;
		if (Item == 0) break;
		Keyword = validate_<Datatype>(Item);
emptynewObject.Insert_an_item_iteratively(Item);
	}
	std::cout << "\n\n\tAfter Insertion  emptynewObject 's data is " <<
emptynewObject;
	std::cout << "\n\tDelete two elements from an object \n";
int count = 0;
	for (count; count < 2; count++) {
		Datatype Number;
		std::cout << "\n\n\tPass the number here to delete : "; std::cin >>
Number;
		Number = validate_<Datatype>(Number);
		emptynewObject.Delete_an_item_iteratively(Number);
	}
	std::cout << "\n\n\tAfter Deletion : " << emptynewObject;
std::cout << "\n*****************************  End of The  Testing 2
*****************************" << std::endl;
	Test1();
}
```