Ex. No. 10	IMPLEMENTATION OF BINARY SEARCH TREE
01-10-2017	

Question:

To implement the working of implementation of binary search tree.

Algorithm:

1. Start.
2. Create struct treenode and declare attributes treenode *left, int data, treenode *right and initialize as *head as null.
3. Create function insert2(), set, if(temp→data<monk→data) and if (monk→left==null), store the temp at left.
4. Else insert2(temp, monk→left)
5. Set, else if (temp→data<monk→data)
6. set, if (monk→right=null), store the temp at right.
7. Else insert2(temp, monk->right)
8. create another insert() function. In that, create a temp pointer of treenode using new
9. set left & right pointer null
10. insert data at info.
11. Set if head==null, then head =temp
12. else insert2(temp,monk).
13. Create show function, preorder(), print in the order data, left & right and for inorder(), print in the order left, data & right and postorder(), print in the order left , right & data.

14. In the main() function, create switch case to invoke the required functions.
15. Perform the necessary operations.
16. Display the result.
17. End.

Program:

```
//IMPLEMENTATION OF BINARY SEARCH TREE
#include <iostream>
using namespace std;
struct treenode
{
	treenode *left;
	int data;
	treenode *right;
}*head=NULL;
void Insert2(treenode *temp, treenode *monk)
{
	if(temp->data<monk->data)
	{
		if(monk->left==NULL)
		{
			monk->left=temp;
```

```
                }
                else
                {
                        Insert2(temp,monk->left);
                }}
else if(temp->data>monk->data)
        {
                if(monk->right==NULL)
                {
                        monk->right=temp;
                }
                else
                {
                        Insert2(temp,monk->right);
                }
        }
}

void insert(int item)
{
        treenode *temp;
        temp=new treenode;
```

```
        temp->left=NULL;
        temp->right=NULL;
        temp->data=item;
        if(head==NULL)
        {
                head=temp;
        }
        else
        {
                treenode *monk=head;
                Insert2(temp,monk);
        }
}
void inorder(treenode *monk)
{
        if(monk!=NULL)
        {
                inorder(monk->left);
                cout<<" "<<monk->data<<" ";
                inorder(monk->right);
        }
        return;
```

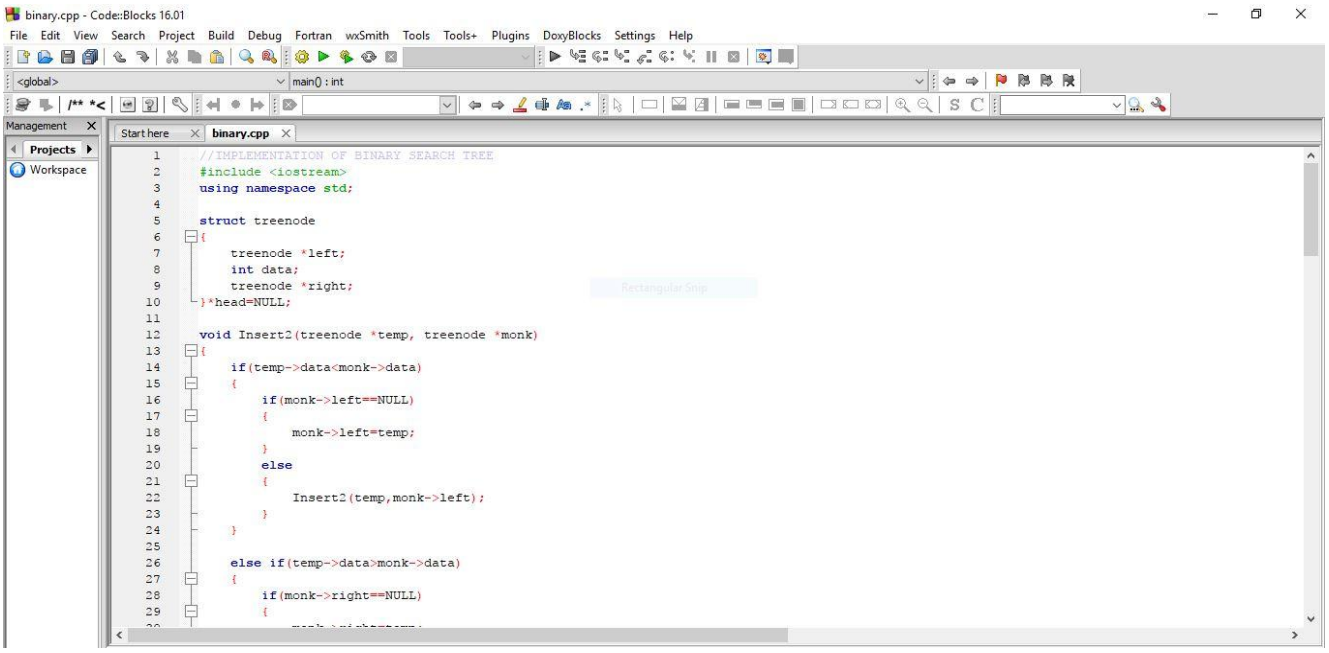
```
}
void preorder(treenode *monk)
{
        if(monk!=NULL)
        {
                cout<<" "<<monk->data<<" ";
                preorder(monk->left);
                preorder(monk->right);
        }
        return;
}
void postorder(treenode *monk)
{
        if(monk!=NULL)
        {
                postorder(monk->left);
                postorder(monk->right);
                cout<<" "<<monk->data<<" ";
        }
        return;
}
int main()
```

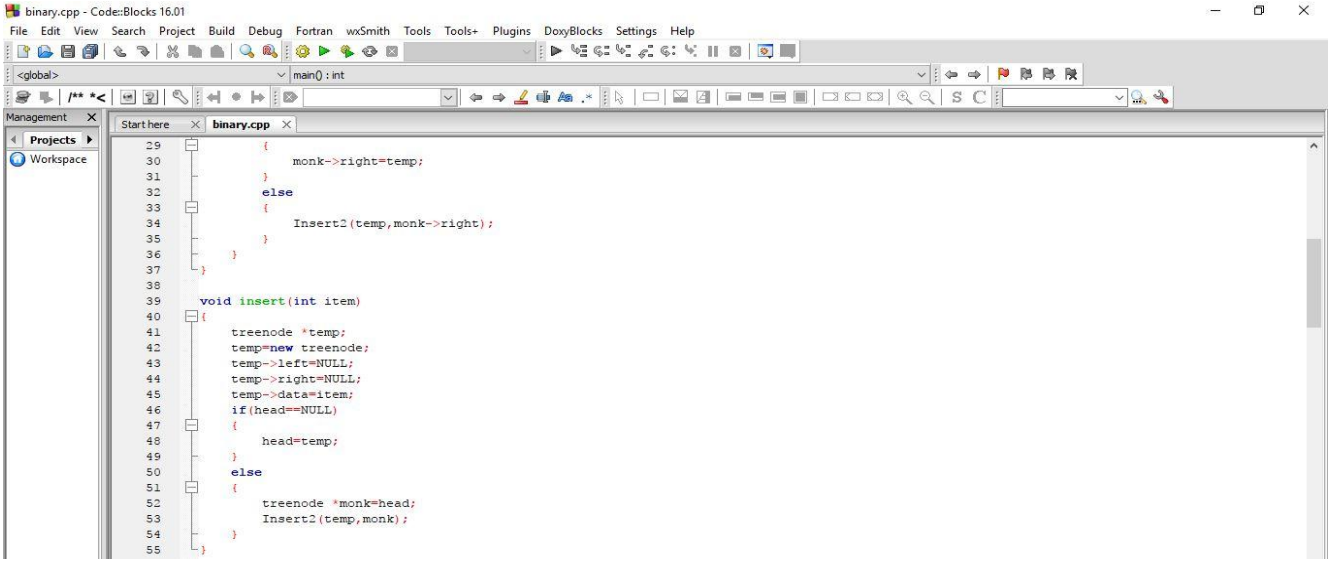
```
{
        int x,n,d;
        while(1)
        {
      prev:
       cout<<"\n To do...............";
       cout<<"\n 1.Insert\n 2.Show\n 3.Exit     ";
        cout<<"\n\n Enter your choice: ";
        cin>>n;
        switch(n)
        {
case 1:         cout<<"\n Enter the data: ";
                cin>>d;
                insert(d);
                break;
case 2:         while(1)
                {
                cout<<"\n\n To do...............";
                cout<<"\n 1.Pre order\n 2.In order\n 3.Post order \n 4.Return      ";
                cout<<"\n\n Enter your  choice: ";
                cin>>x;
                switch(x)
```

```
{
case 1:
cout<<"\n Preorder Traversal:\n";
preorder(head);
        break;
        case 2:
         cout<<"\n Inorder Traversal:\n";
          inorder(head);
        break;
       case 3:
        cout<<"\n Postorder Traversal:\n";
       postorder(head);
       break;
        case 4:
       goto prev;
        default:
        cout<<"\n Invalid order"<<endl;
                                }
                        }
        break;
        case 3:
       return 0;
```

```
                default:        cout<<"Invalid order"<<endl;
            }
        }
}
```

Output:





binary.cpp - Code::Blocks 16.01

```
void inorder(treenode *monk)
{
    if(monk!=NULL)
    {
        inorder(monk->left);
        cout<<" "<<monk->data<<" ";
        inorder(monk->right);
    }
    return;
}

void preorder(treenode *monk)
{
    if(monk!=NULL)
    {
        cout<<" "<<monk->data<<" ";
        preorder(monk->left);
        preorder(monk->right);
    }
    return;
}

void postorder(treenode *monk)
{
    if(monk!=NULL)
    {
        postorder(monk->left);
        postorder(monk->right);
        cout<<" "<<monk->data<<" ";
```

binary.cpp - Code::Blocks 16.01

```
    }
    return;
}

int main()
{
    int x,n,d;
    while(1)
    {
        prev:
        cout<<"\n To do...............";
        cout<<"\n 1.Insert\n 2.Show\n 3.Exit    ";
        cout<<"\n\n Enter your choice: ";
        cin>>n;
        switch(n)
        {
            case 1: cout<<"\n Enter the data: ";
                    cin>>d;
                    insert(d);
                break;
            case 2: while(1)
                    {
                        cout<<"\n\n To do...............";
                        cout<<"\n 1.Pre order\n 2.In order\n 3.Post order \n 4.Return   ";
                        cout<<"\n\n Enter your  choice: ";
                        cin>>x;
                        switch(x)
                        {
                            case 1:
```

binary.cpp - Code::Blocks 16.01

```
                    cout<<"\n\n Enter your  choice: ";
                    cin>>x;
                    switch(x)
                    {
                        case 1:
                            cout<<"\n Preorder Traversal:\n";
                            preorder(head);
                            break;
                        case 2:
                            cout<<"\n Inorder Traversal:\n";
                             inorder(head);
                            break;
                        case 3:
                                cout<<"\n Postorder Traversal:\n";
                                postorder(head);
                                break;
                        case 4:
                            goto prev;
                        default:
                             cout<<"\n Invalid order"<<endl;
                    }
                }
            break;
        case 3: return 0;
        default:    cout<<"Invalid order"<<endl;
    }
}
}
```

"C:\Users\JUSTIN PAUL\Desktop\binary.exe"

```
To do...............
1.Insert
2.Show
3.Exit

Enter your choice: 1

Enter the data: 10

To do...............
1.Insert
2.Show
3.Exit

Enter your choice: 1

Enter the data: 20

To do...............
1.Insert
2.Show
3.Exit

Enter your choice: 1

Enter the data: 33

To do...............
1.Insert
2.Show
3.Exit

Enter your choice: 1

Enter the data: 43
```

```
"C:\Users\JUSTIN PAUL\Desktop\binary.exe"
 To do...............
 1.Insert
 2.Show
 3.Exit

 Enter your choice: 2


 To do...............
 1.Pre order
 2.In order
 3.Post order
 4.Return

 Enter your  choice: 1

 Preorder Traversal:
 10  20  33  43

 To do...............
 1.Pre order
 2.In order
 3.Post order
 4.Return

 Enter your  choice:
2

 Inorder Traversal:
 10  20  33  43

 To do...............
 1.Pre order
 2.In order
 3.Post order
 4.Return

 Enter your  choice: 3

 Postorder Traversal:
 43  33  20  10
```

VIDEO URL:

https://youtu.be/AftsqAUyA2Q

RESULT:

The program of implementation of binary search tree is implemented successfully and the output is verified.