

```
8
9 #include<stdio.h>
10 #include<stdlib.h>
11
12 typedef struct node
13 {
14     int data;
15     struct node *left;
16     struct node *right;
17 }Node;
18 void tree();
19 Node * create();
20 Node *insert(Node *,Node *);
21 void traverse();
22 void preOrder(Node *);
23 void inOrder(Node *);
24 void postOrder(Node *);
25 void display(Node *,int);
26
27 Node *root;
28
29 int main()
30 {
31     tree();
32     return 0;
33 }
34 void tree()
35 {
36     int choice;
37     printf("\n <--Binary Search Tree-->\n 1.Insert Element\n 2.Traverse-All methods\n 3.Display BST\n 4.Exit\n Choice: ");
```

```
main.c
37 printf("\n <--Binary Search Tree-->\n 1.Insert Element\n 2.Traverse-All methods\n 3.Display BST\n 4.Exit\n Choice: ");
38 scanf("%d",&choice);
39 switch(choice)
40 {
41     case 1: insert(root,create()); break;
42     case 2: traverse(); break;
43     case 3: if(root==NULL)
44             printf("\n Tree is Empty!");
45             else
46             display(root,0);
47             break;
48     case 4: exit(0);break;
49     default: printf("\n Error Choice !\n ");
50             tree();
51 }
52 tree();
53 }
54 Node * create()
55 {
56     Node* newnode=(Node *)malloc(sizeof(Node));
57     printf("\n Enter the Element: ");
58     scanf("%d",&newnode->data);
59     newnode->left=NULL;
60     newnode->right=NULL;
61     return newnode;
62 }
63 Node * insert(Node *Root,Node *newNode)
64 {
65     if(root==NULL)
66     {
```


onlinegdb.com/online_c_compiler

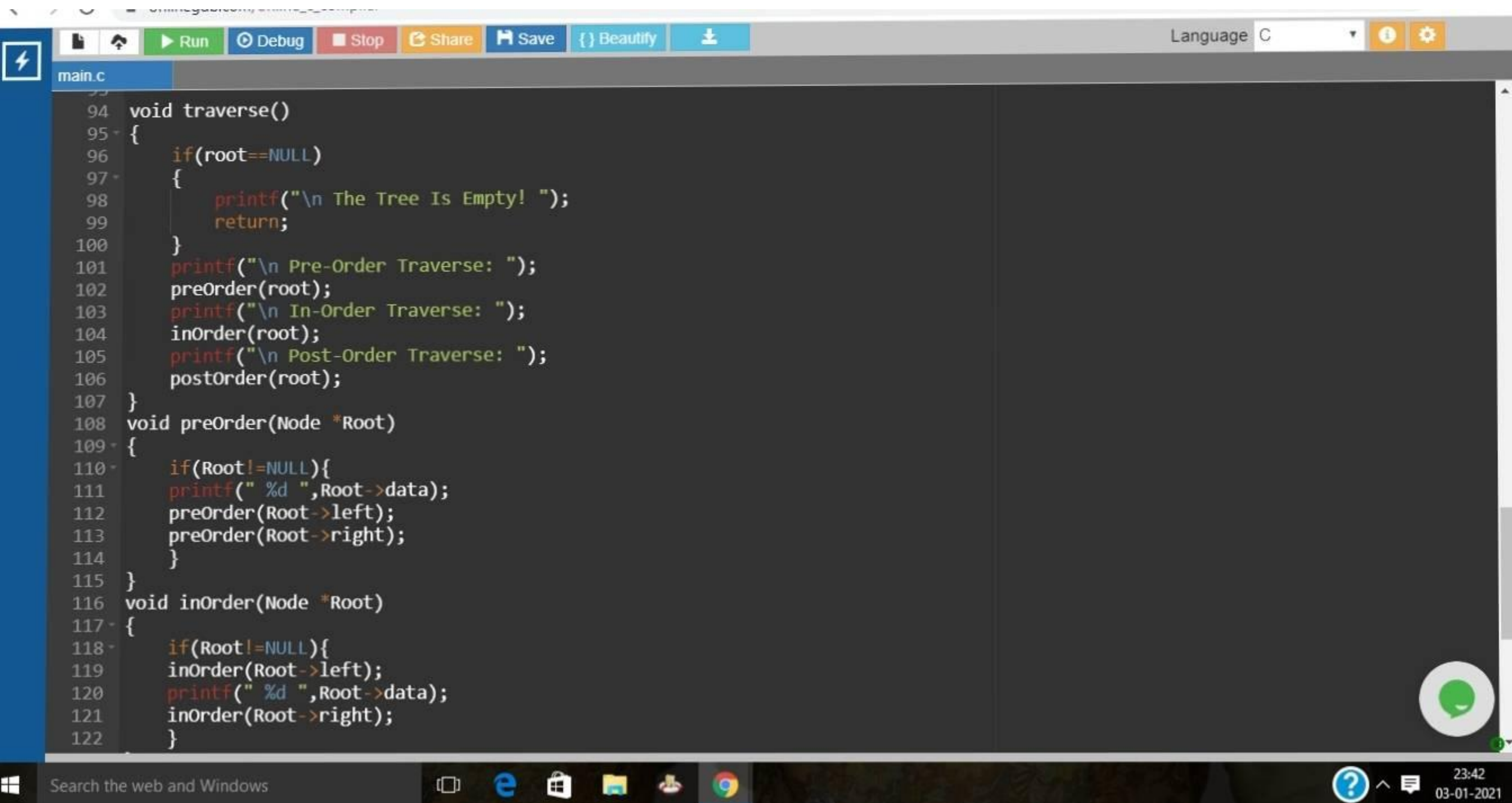
main.c

```
65 if(root==NULL)
66 {
67     root=newNode;
68     printf("\n Root Node Created ");
69 }
70 else
71 {
72     if(newNode->data>Root->data)
73     {
74         if(Root->right==NULL)
75         {
76             Root->right=newNode;
77         }
78         else
79             insert(Root->right,newNode);
80     }
81     else
82     if(newNode->data<Root->data)
83     {
84         if(Root->left==NULL)
85         {
86             Root->left=newNode;
87         }
88         else
89             insert(Root->left,newNode);
90     }
91 }
92 }
93 }
```

Language C

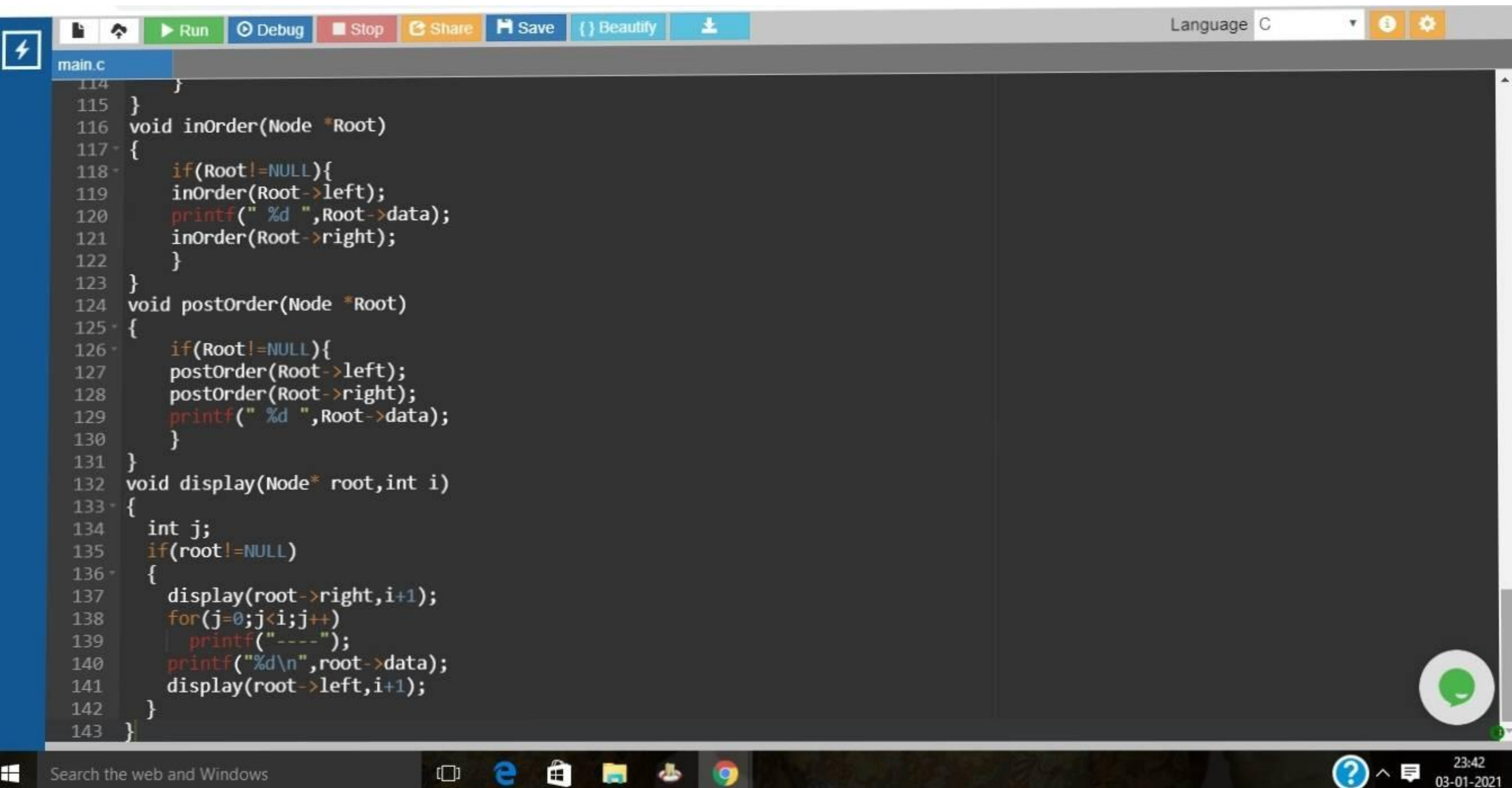
Search the web and Windows

23:41 03-01-2021



The image shows a screenshot of a code editor window with a dark theme. The editor is displaying C code for tree traversal. The code is organized into three functions: `traverse()`, `preOrder()`, and `inOrder()`. The `traverse()` function checks if the root is NULL and prints a message if it is. It then calls `preOrder()`, `inOrder()`, and `postOrder()` on the root. The `preOrder()` function prints the data of the root node and recursively calls itself on the left and right children. The `inOrder()` function recursively calls itself on the left child, prints the data of the root node, and then recursively calls itself on the right child. The code is written in a C file named `main.c`. The editor has a toolbar at the top with buttons for Run, Debug, Stop, Share, Save, Beautify, and a download icon. The language is set to C. The Windows taskbar is visible at the bottom, showing the search bar and several application icons.

```
94 void traverse()
95 {
96     if(root==NULL)
97     {
98         printf("\n The Tree Is Empty! ");
99         return;
100     }
101     printf("\n Pre-Order Traverse: ");
102     preOrder(root);
103     printf("\n In-Order Traverse: ");
104     inOrder(root);
105     printf("\n Post-Order Traverse: ");
106     postOrder(root);
107 }
108 void preOrder(Node *Root)
109 {
110     if(Root!=NULL){
111         printf(" %d ",Root->data);
112         preOrder(Root->left);
113         preOrder(Root->right);
114     }
115 }
116 void inOrder(Node *Root)
117 {
118     if(Root!=NULL){
119         inOrder(Root->left);
120         printf(" %d ",Root->data);
121         inOrder(Root->right);
122     }
```

```
114 }
115 }
116 void inOrder(Node *Root)
117 {
118     if(Root!=NULL){
119         inOrder(Root->left);
120         printf(" %d ",Root->data);
121         inOrder(Root->right);
122     }
123 }
124 void postOrder(Node *Root)
125 {
126     if(Root!=NULL){
127         postOrder(Root->left);
128         postOrder(Root->right);
129         printf(" %d ",Root->data);
130     }
131 }
132 void display(Node* root,int i)
133 {
134     int j;
135     if(root!=NULL)
136     {
137         display(root->right,i+1);
138         for(j=0;j<i;j++)
139             printf("----");
140         printf("%d\n",root->data);
141         display(root->left,i+1);
142     }
143 }
```

main.c

Language C

Search the web and Windows

23:42 03-01-2021

```
input
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

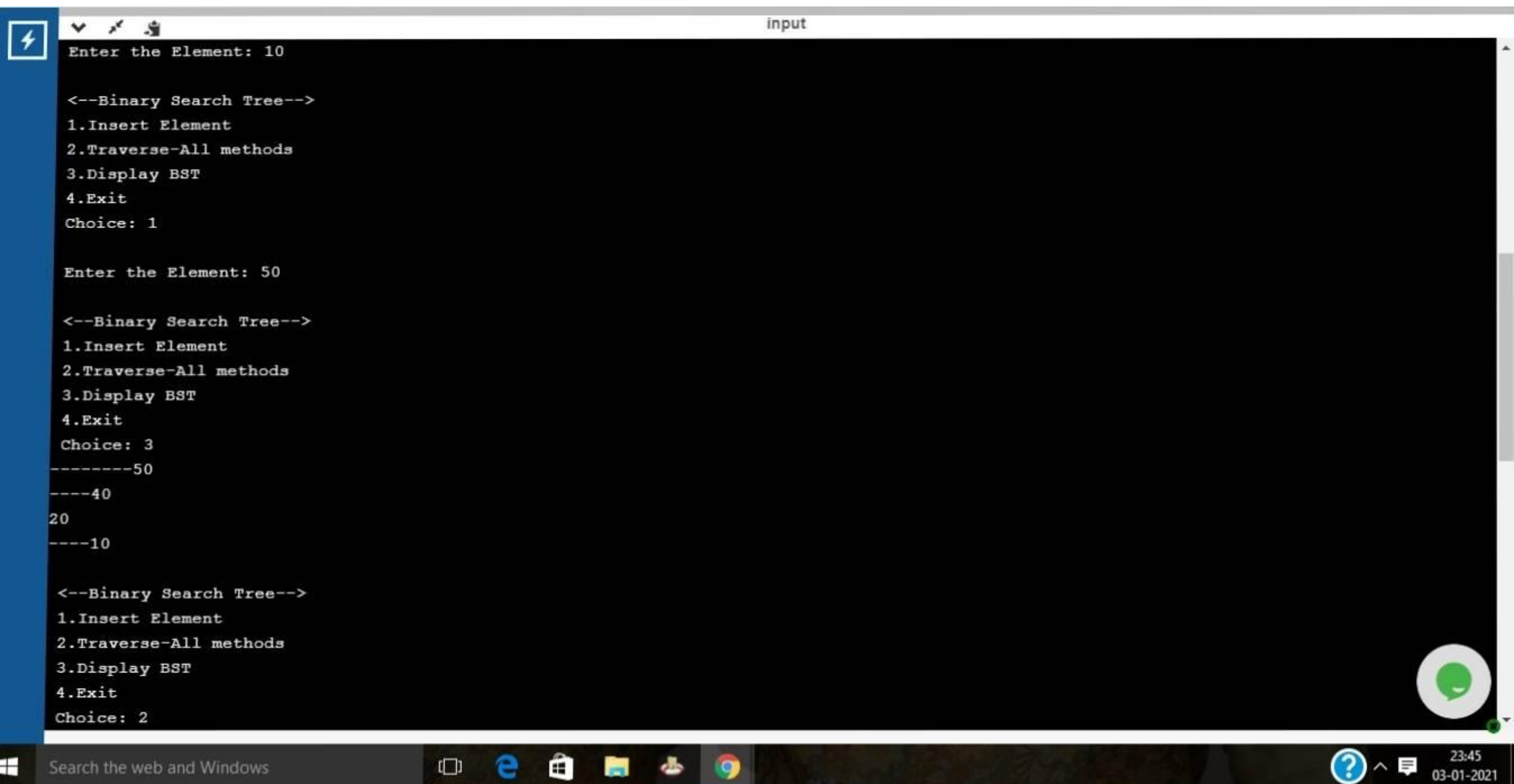
Enter the Element: 20

Root Node Created
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 40

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 10
```



```
input
4.Exit
Choice: 2

Pre-Order Traverse: 20 10 40 50
In-Order Traverse: 10 20 40 50
Post-Order Traverse: 10 50 40 20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 80

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 2

Pre-Order Traverse: 20 10 40 50 80
In-Order Traverse: 10 20 40 50 80
Post-Order Traverse: 10 80 50 40 20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
```



```
input
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 1

Enter the Element: 80

<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 2

Pre-Order Traverse:  20  10  40  50  80
In-Order Traverse:   10  20  40  50  80
Post-Order Traverse:  10  80  50  40  20
<--Binary Search Tree-->
1.Insert Element
2.Traverse-All methods
3.Display BST
4.Exit
Choice: 4

...Program finished with exit code 0
Press ENTER to exit console.
```