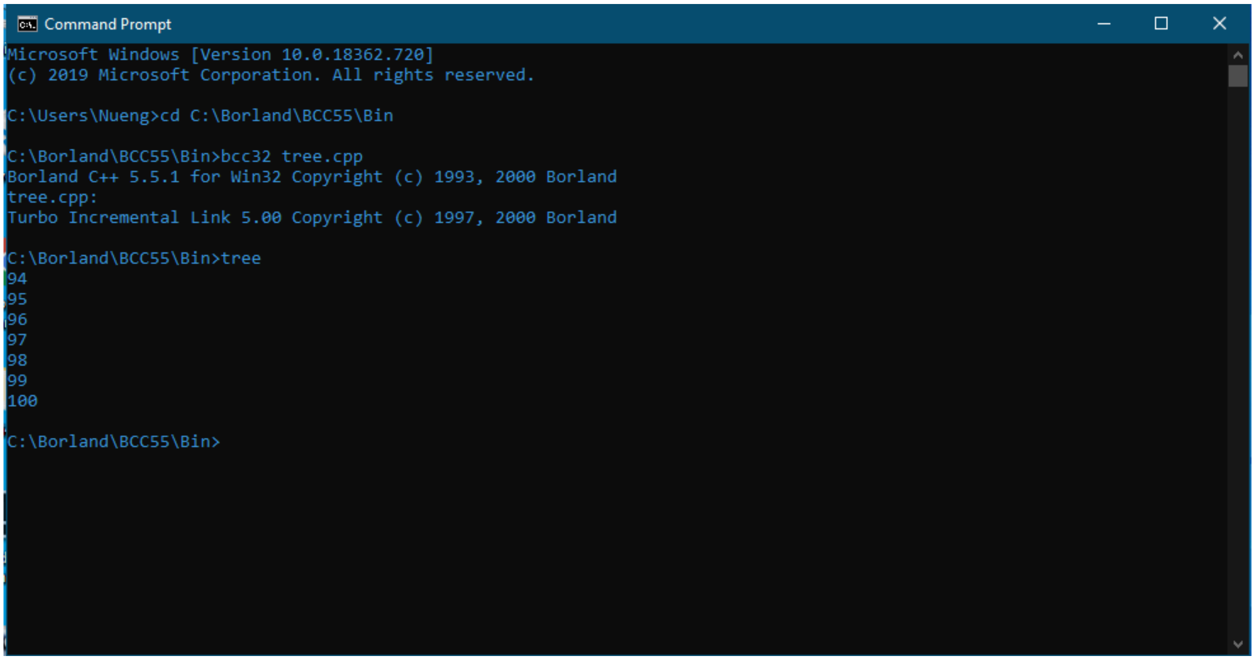

```
Microsoft Windows [Version 10.0.18362.720]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Nueng>cd C:\Borland\BCC55\Bin

C:\Borland\BCC55\Bin>bcc32 tree.cpp
Borland C++ 5.5.1 for Win32 Copyright (c) 1993, 2000 Borland
tree.cpp:
Turbo Incremental Link 5.00 Copyright (c) 1997, 2000 Borland

C:\Borland\BCC55\Bin>tree
94
95
96
97
98
99
100

C:\Borland\BCC55\Bin>
```

```
// C++ program to demonstrate insertion
// in a Nueng recursively.
#include <iostream>
using namespace std;

class Nueng
{
    int data;
    Nueng *left, *right;

    public:

    // Default constructor.
    Nueng();

    // Parameterized constructor.
    Nueng(int);
```

```
   // Insert function.
   Nueng* Insert (Nueng*, int);

   // Inorder traversal.
   void Inorder (Nueng*);
};

// Default Constructor definition.
Nueng:: Nueng() : data(0), left(NULL), right(NULL){}

// Parameterized Constructor definition.
Nueng:: Nueng(int value)
{
   data = value;
   left = right = NULL;
}

// Insert function definition.
Nueng* Nueng :: Insert (Nueng *root, int value)
{
   if(!root)
   {
      // Insert the first node, if root is NULL.
      return new Nueng(value);
   }

   // Insert data.
   if(value > root->data)
   {
      // Insert right node data, if the 'value'
      // to be inserted is greater than 'root' node data.

      // Process right nodes.
      root->right = Insert(root->right, value);
   }
   else
   {
      // Insert left node data, if the 'value'
      // to be inserted is greater than 'root' node data.

      // Process left nodes.
      root->left = Insert(root->left, value);
   }

   // Return 'root' node, after insertion.
   return root;
}

// Inorder traversal function.
// This gives data in sorted order.
void Nueng :: Inorder(Nueng *root)
{
   if(!root)
   {
      return;
   }
   Inorder(root->left);
   cout << root->data << endl;
   Inorder(root->right);
}
```

```
// Driver code
int main()
{
    Nueng b, *root = NULL;
    root = b.Insert(root, 94);
    b.Insert(root, 100);
    b.Insert(root, 99);
    b.Insert(root, 98);
    b.Insert(root, 97);
    b.Insert(root, 96);
    b.Insert(root, 95);

    b.Inorder(root);
    return 0;
}
```