**4. Implementing TDD on c++ mini project**

*Problem statement:* *Create Binary search tree in c++ using TDD*

**Steps in TDD(Test Driven Development)**

**Step 1:** Write enough failing test code (Including compile time and runtime)

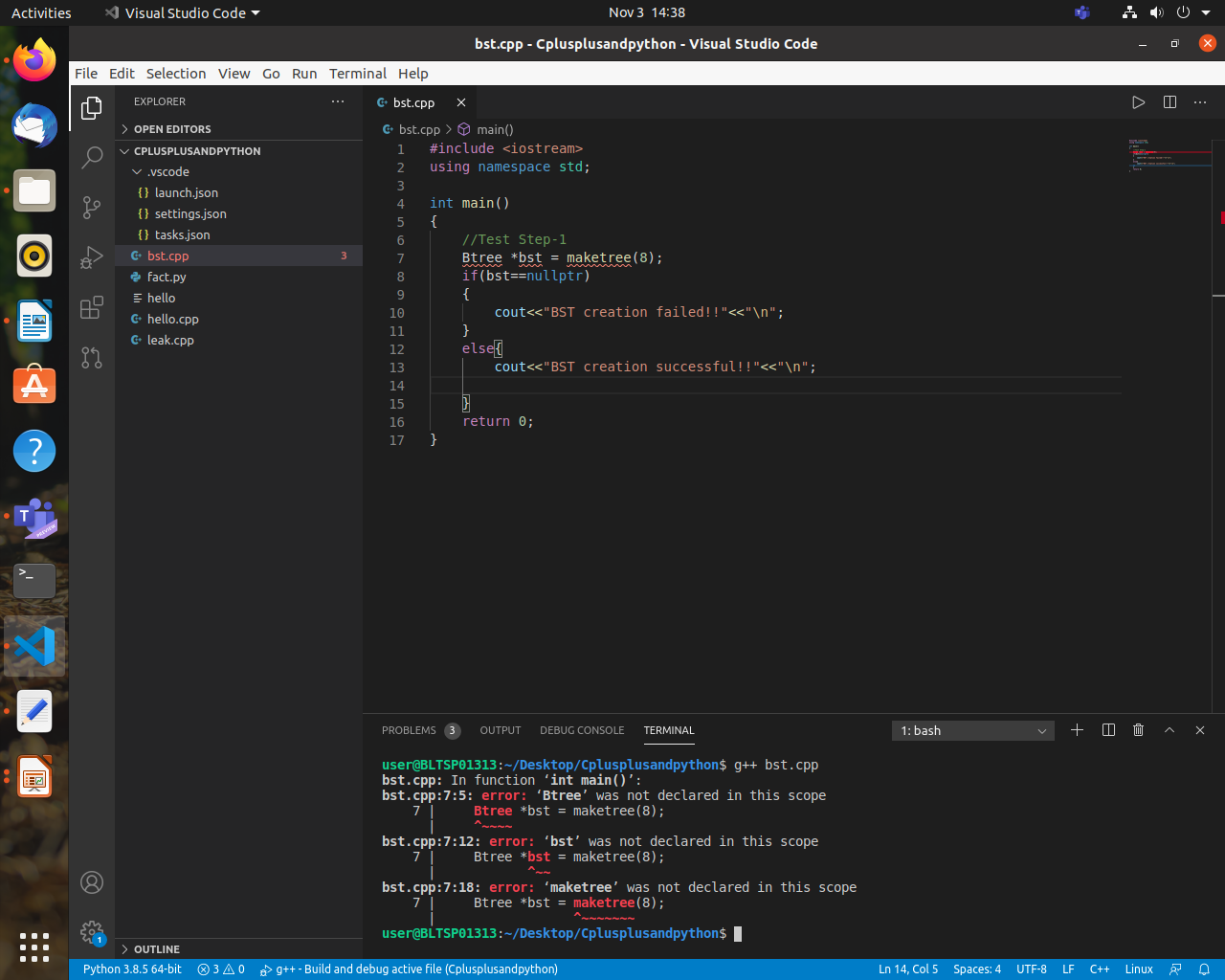
**Step 2:** Write production code to pass those failing Tests

**Step 3:** Refactor production code and verify the same with the existing test.

**Step 4:** Repeat (Go to Step 1) until all tests are passed and code is refactored.

Now let us write some failing test code:

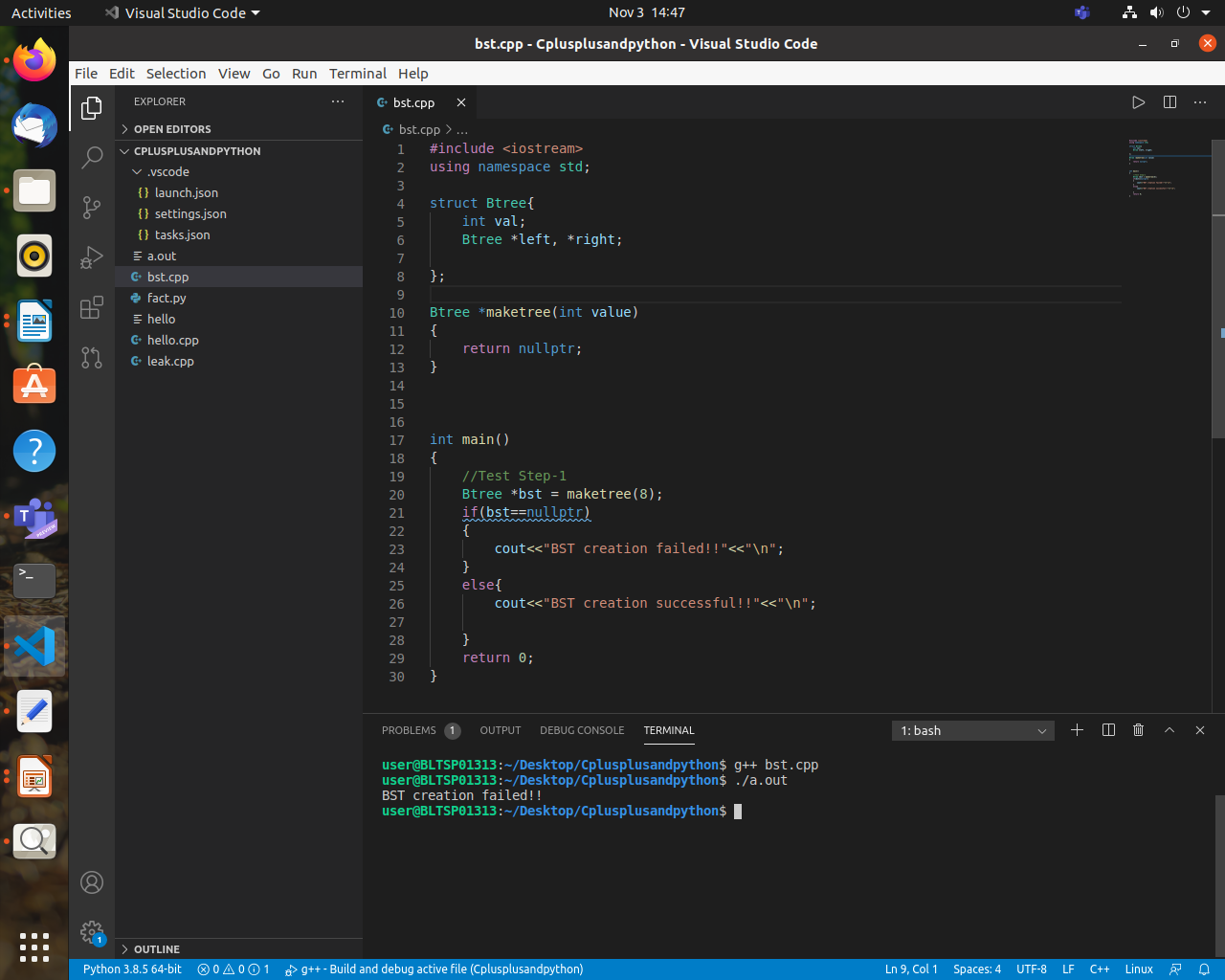
**Step1:**  Here we write test code first, which is bound to fail as there is no production code.



**Figure 1:** Test code 1 (Compile Time Error)

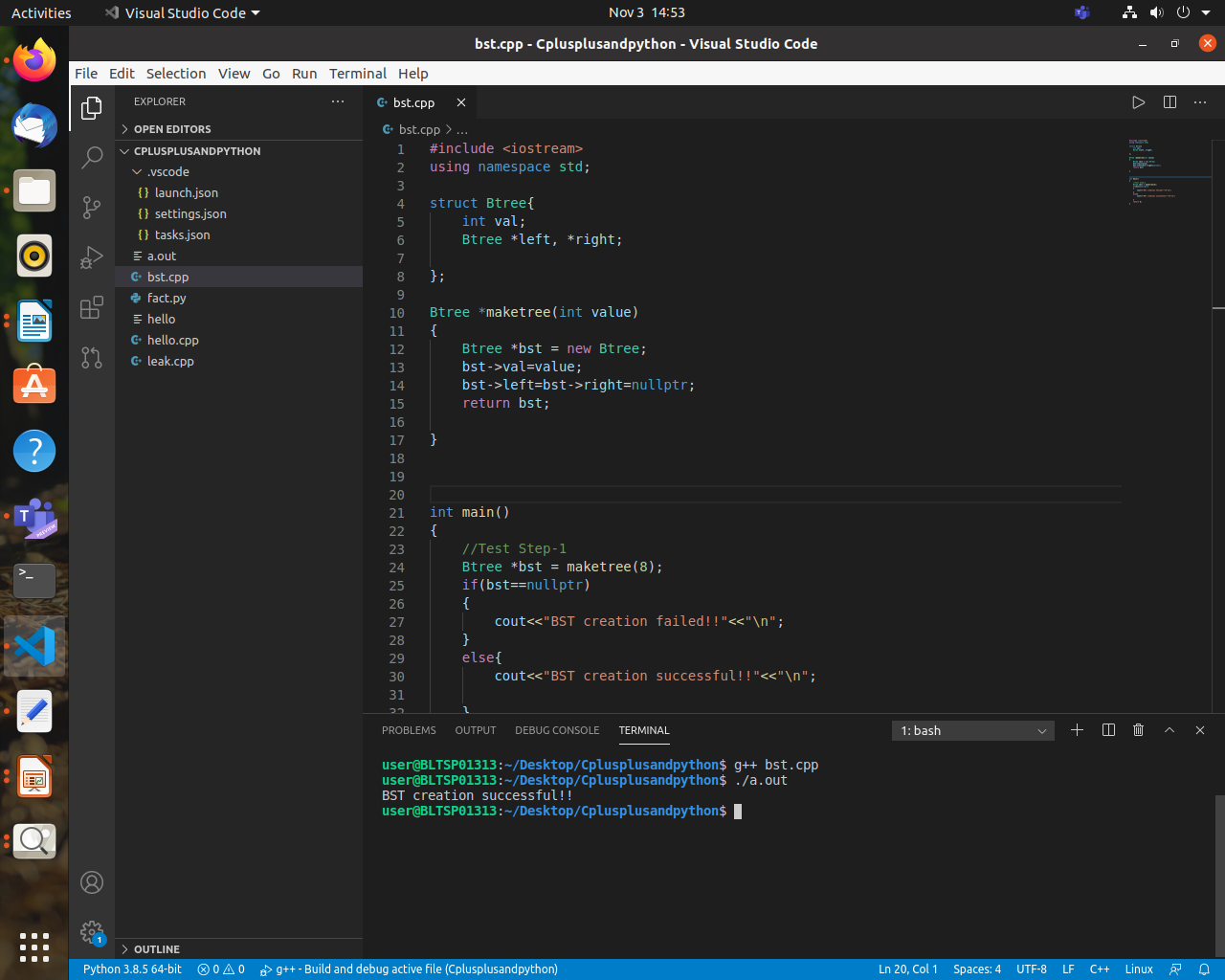
We can see that we have written test code which are failing as Btree and maketree are not defined. Here we are having compile time errors.

**Step2:** Now we shall write some production code to make sure we pass test code



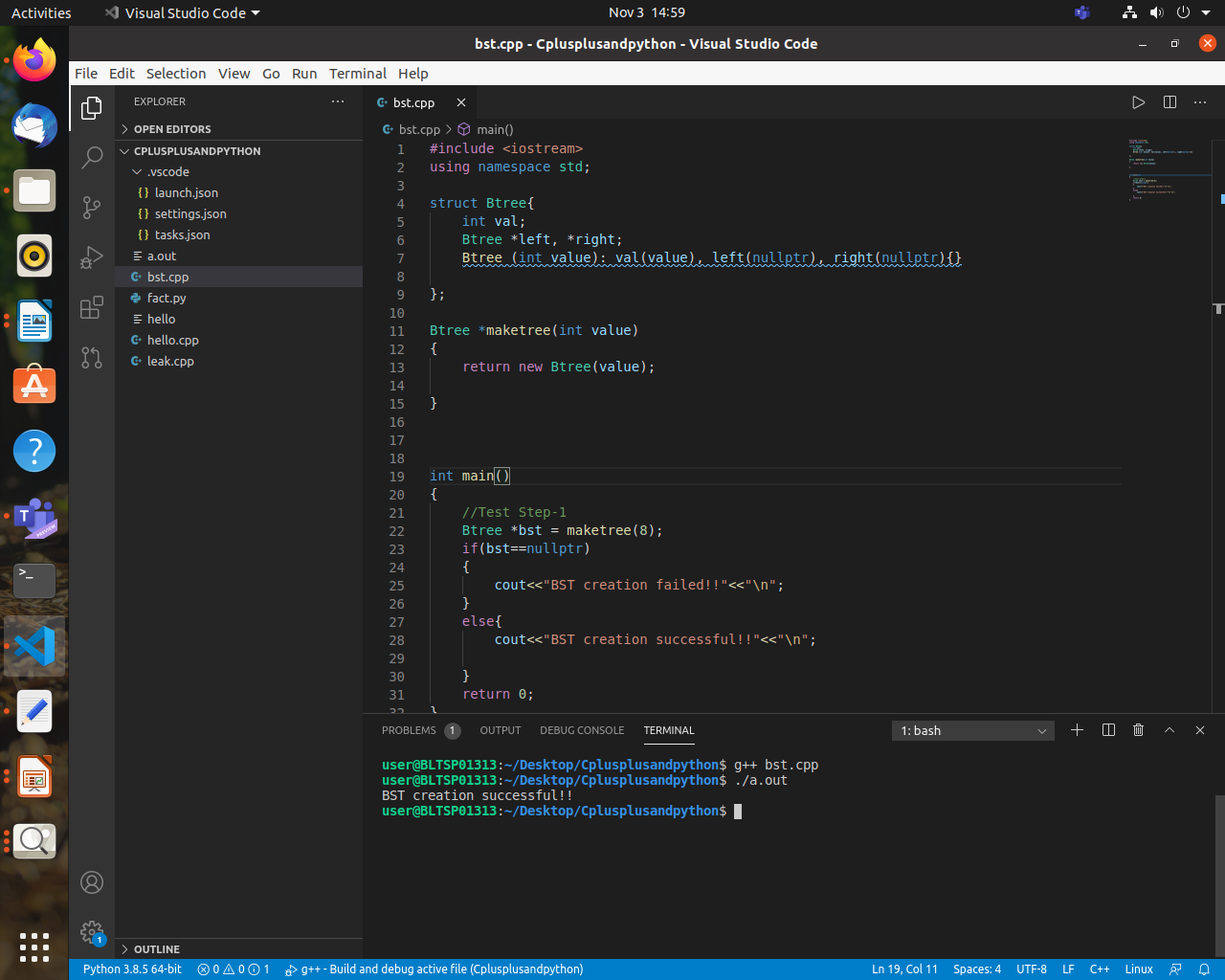
**Figure 2:** Test code1 along with Production code (RunTime Error)

We have written sufficient production code to successfully compile the code, but here we have runtime error. So we have come from compile time error to runtime errors.

  **Figure 3:** Test Code1 along with Production Code (working successfully)

We have executed the code successfully and passed the test code. We have solved runtime errors.

**Step 3:** We have to refactor the code(adding enhancements to the existing code), and we check if code successfully works:

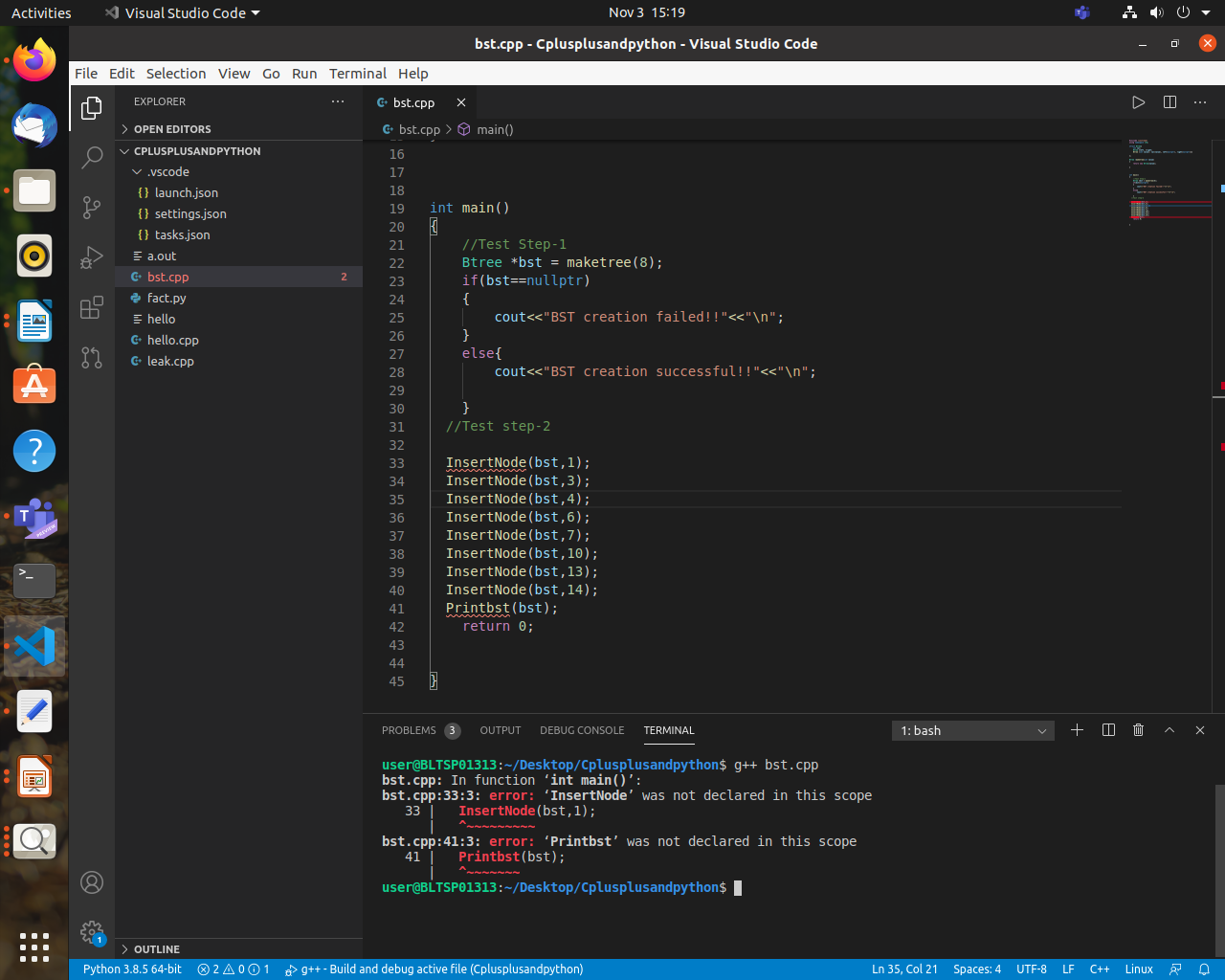
 **Figure 4**: Refactored Production Code with Test code 1 (Working Successfully)

We can see that there are some changes made to production code to make it look better and some extra features are added. Code successfully compiled and then executed without any runtime errors.

After 3 steps are over, we are going to repeat these 3 steps again

***Step 1(2nd iteration):***

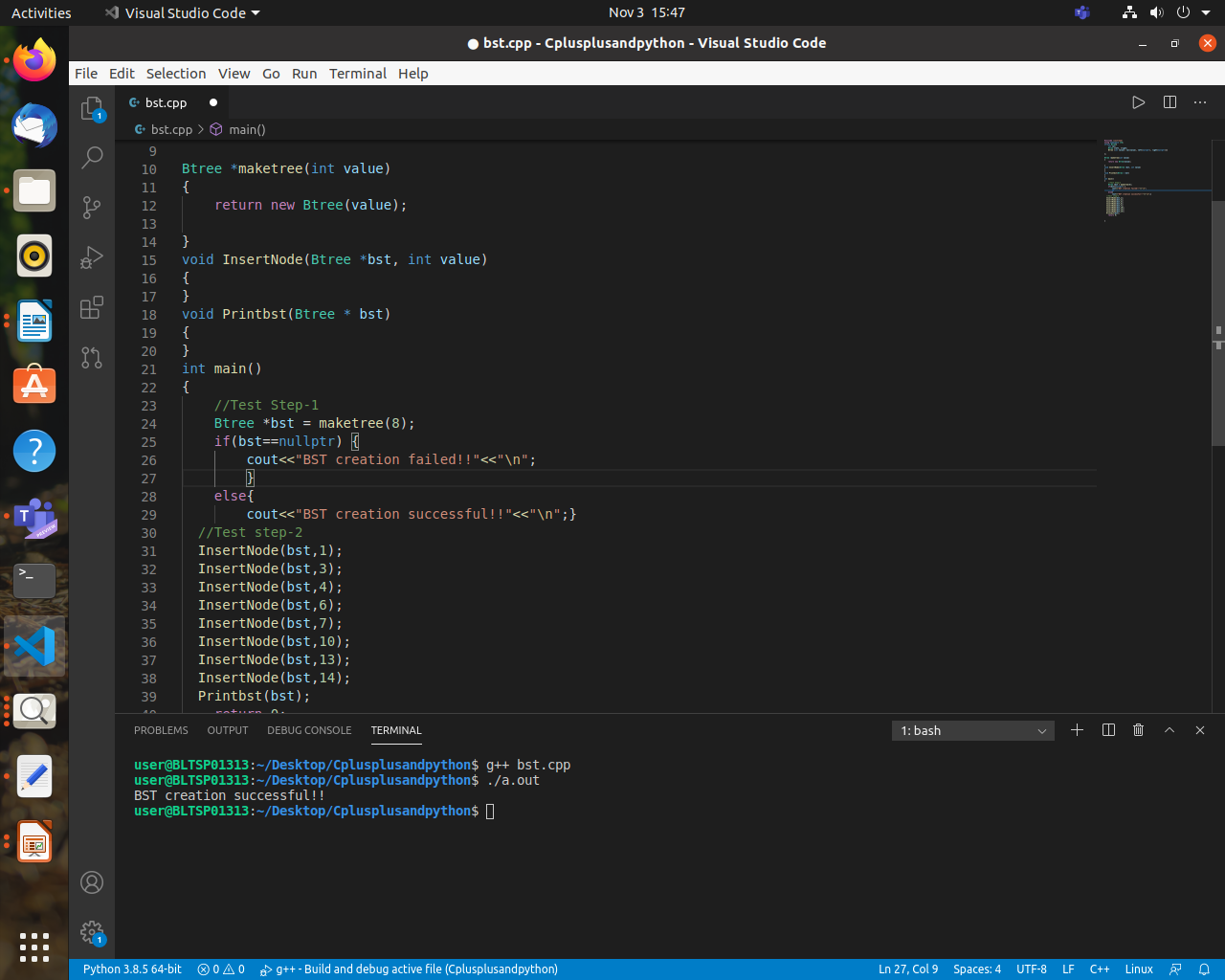
We have written test code and it has failed because there is no function defined as InsertNode.So we have to write production code to pass the test code.



**Figure 5**: Test Code 2 (Compile Time Error)

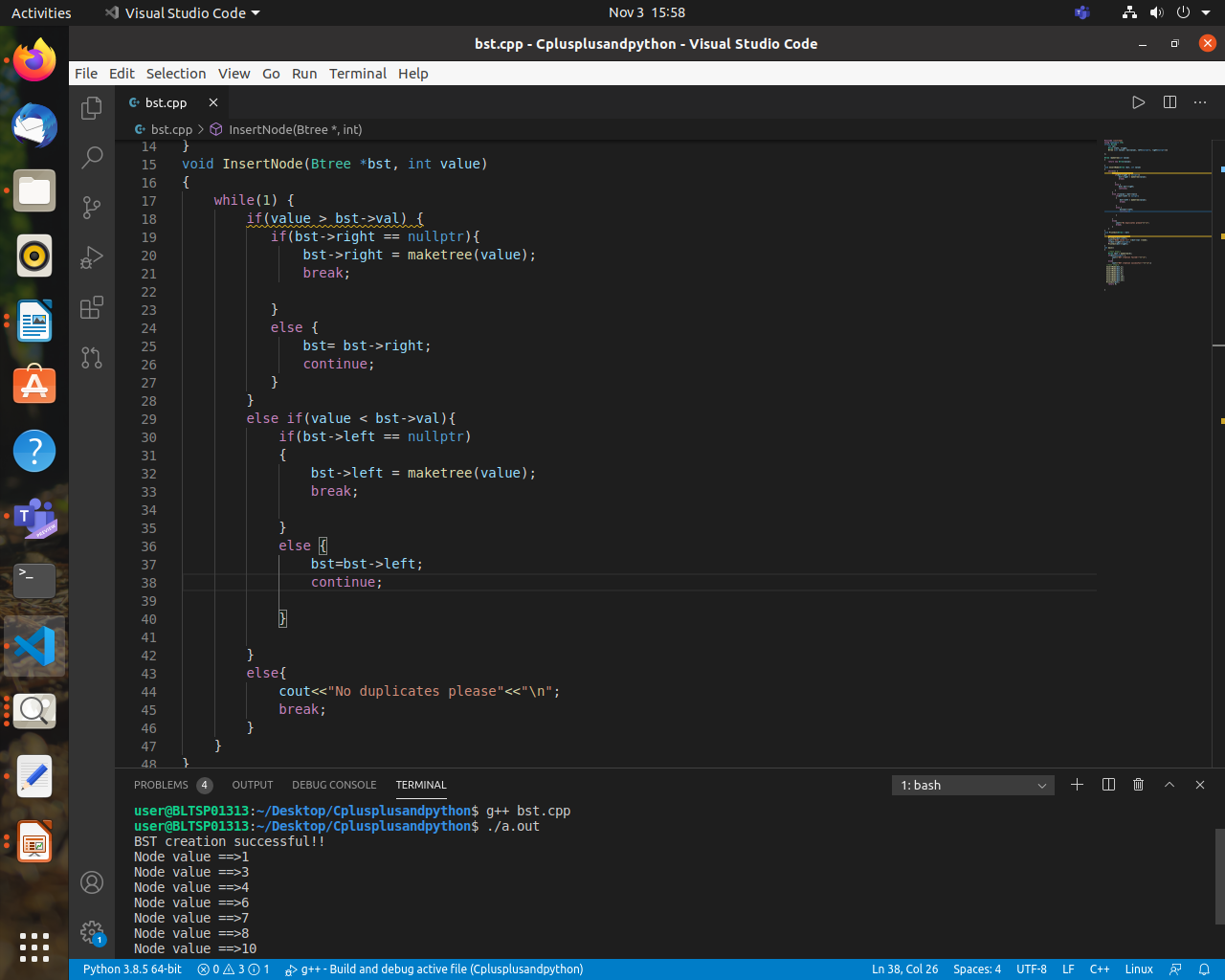
***Step 2 (2nd iteration):***

Now we will write sufficient production code so that it passes the test code and executes successfully.



**Figure 6:** Test Code2 along with Production Code (working successfully)

***Step 3 (2nd iteration):*** Now we will refactor the code, i.e add some new features or enhancements to make the code look better and have better functionalities.



**Figure 7**: Refactored Production Code with Test code 2 (Working Successfully)

We have refactored the code, and it compiles and then executes successfully.

**Summary:**

This activity demonstrates Test Driven Development, where we write test code first and then fail it, and then write sufficient production code to make the test code work and then we will do refactoring which is basically enhancing the code to make it better and the steps are repeated.