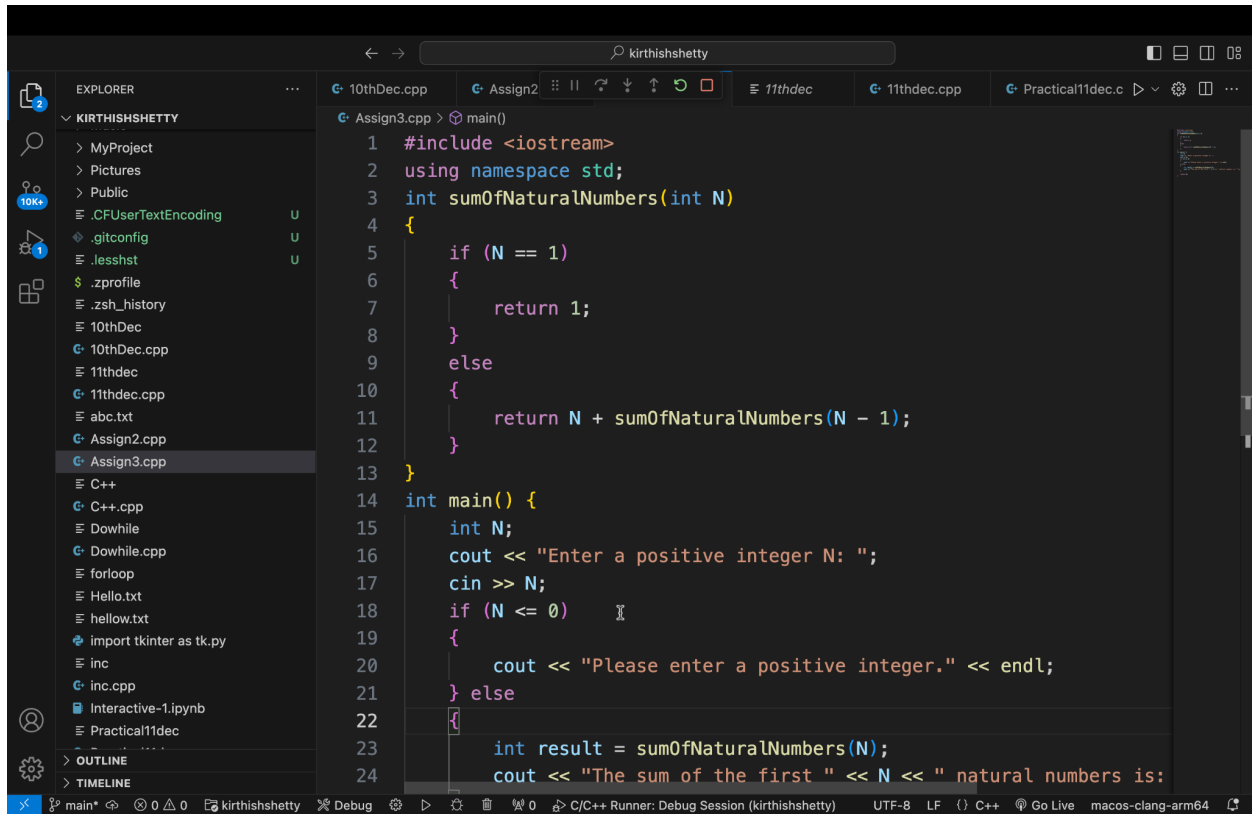


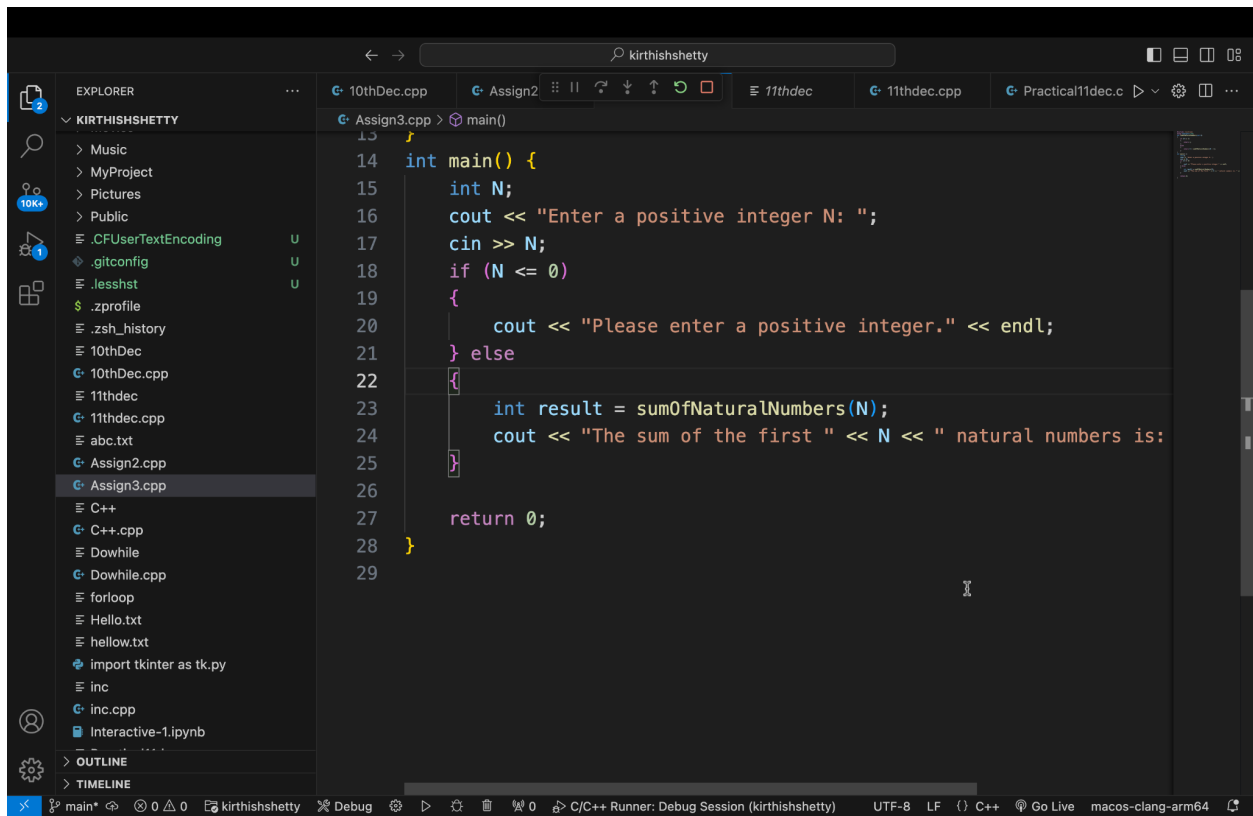
Assignment 3

1. Write a program in C++ to calculate the sum of first N natural numbers using recursion



The screenshot shows a C++ IDE with a file explorer on the left and a code editor in the center. The file explorer shows a project named 'KIRTHISHSHETTY' with various files and folders. The code editor displays the following C++ code:

```
1 #include <iostream>
2 using namespace std;
3 int sumOfNaturalNumbers(int N)
4 {
5     if (N == 1)
6     {
7         return 1;
8     }
9     else
10    {
11        return N + sumOfNaturalNumbers(N - 1);
12    }
13 }
14 int main() {
15     int N;
16     cout << "Enter a positive integer N: ";
17     cin >> N;
18     if (N <= 0)
19     {
20         cout << "Please enter a positive integer." << endl;
21     } else
22     {
23         int result = sumOfNaturalNumbers(N);
24         cout << "The sum of the first " << N << " natural numbers is:"
```



The screenshot shows a C++ IDE with a file explorer on the left and a code editor in the center. The file explorer lists various files and folders, including 'Music', 'MyProject', 'Pictures', 'Public', '.CFUserTextEncoding', '.gitconfig', '.lessht', '.zprofile', '.zsh_history', '10thDec', '10thDec.cpp', '11thdec', '11thdec.cpp', 'abc.txt', 'Assign2.cpp', 'Assign3.cpp', 'C++', 'C++.cpp', 'Dowhile', 'Dowhile.cpp', 'forloop', 'Hello.txt', 'hellow.txt', 'import tkinter as tk.py', 'inc', 'inc.cpp', and 'Interactive-1.ipynb'. The code editor displays the following C++ code:

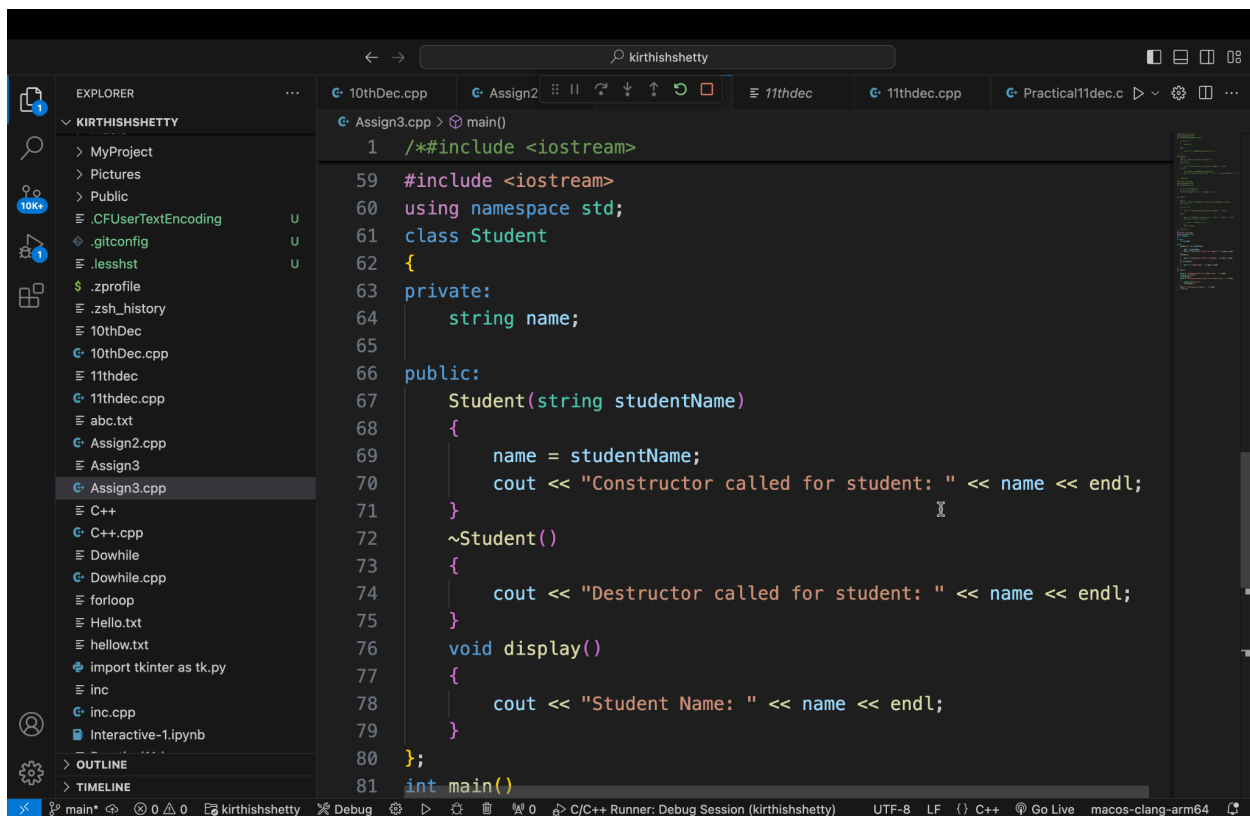
```
14 int main() {  
15     int N;  
16     cout << "Enter a positive integer N: ";  
17     cin >> N;  
18     if (N <= 0)  
19     {  
20         cout << "Please enter a positive integer." << endl;  
21     } else  
22     {  
23         int result = sumOfNaturalNumbers(N);  
24         cout << "The sum of the first " << N << " natural numbers is:  
25     }  
26  
27     return 0;  
28 }  
29
```

2. Write a C++ to find fibonacci series using recursion

```
29 #include <iostream>
30 using namespace std;
31 int fibonacci(int n)
32 {
33     if (n == 0) return 0;
34     if (n == 1) return 1;
35     return fibonacci(n - 1) + fibonacci(n - 2);
36 }
37
38 int main()
39 {
40     int N;
41     cout << "Enter the number of terms in the Fibonacci series: ";
42     cin >> N;
43
44     if (N <= 0)
45     {
46         cout << "Please enter a positive integer." << endl;
47     }
48     else
49     {
50         cout << "Fibonacci series up to " << N << " terms:" << endl;
51         for (int i = 0; i < N; i++)
52         {
```

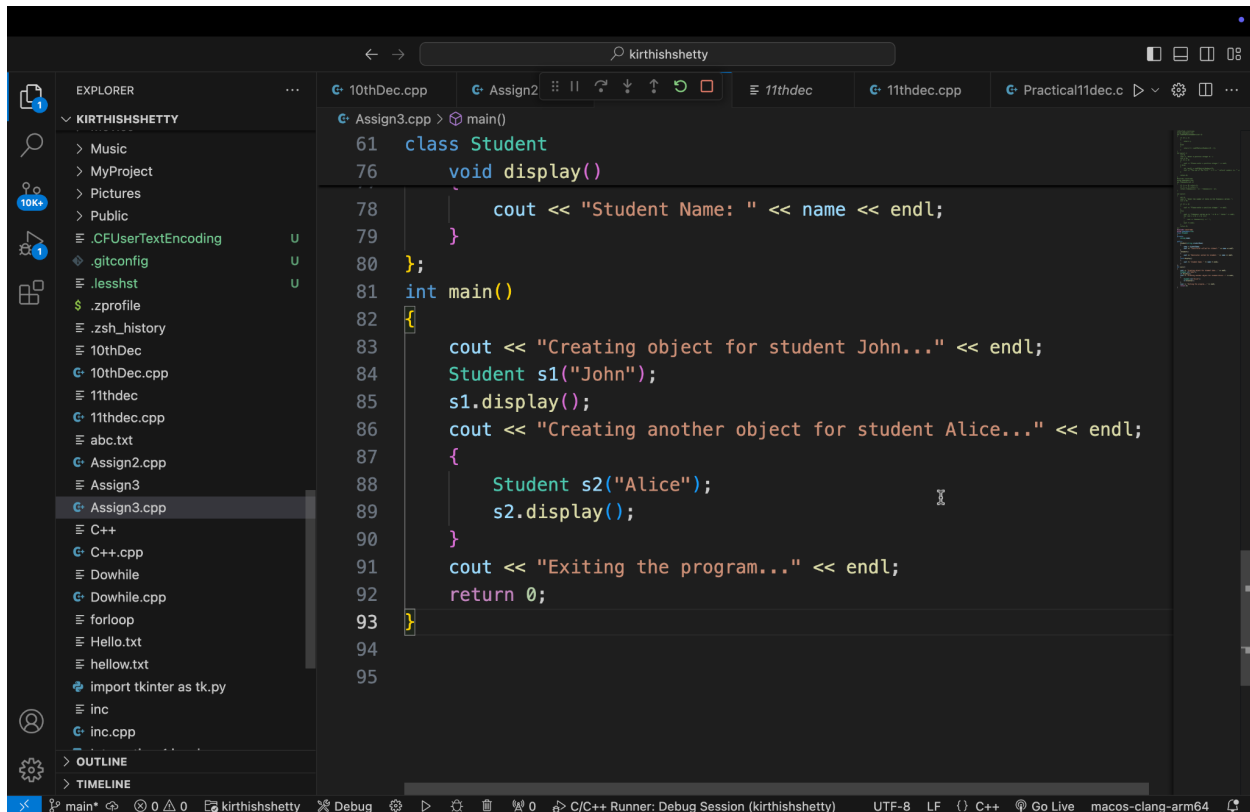
```
38 int main()
44     if (N <= 0)
45     {
46         cout << "Please enter a positive integer." << endl;
47     }
48     else
49     {
50         cout << "Fibonacci series up to " << N << " terms:" << endl;
51         for (int i = 0; i < N; i++)
52         {
53             cout << fibonacci(i) << " ";
54         }
55         cout << endl;
56     }
57     return 0;
58 }
59
```

3.WAP to demonstrate destructor in C++



This screenshot shows the Visual Studio Code editor with the file Explorer on the left displaying a project named 'KIRTHISHSHETTY'. The Explorer lists various files and folders, including 'MyProject', 'Pictures', 'Public', and several source files like '10thDec.cpp', '11thdec.cpp', and 'Assign3.cpp'. The main editor window displays the code for 'Assign3.cpp', which defines a 'Student' class. The class has a private member 'name' of type 'string'. It includes a constructor 'Student(string studentName)' that initializes 'name' and prints a message. It also includes a destructor '~Student()' that prints a message when the object is destroyed. A 'display()' method is also defined to print the student's name. The 'main()' function is partially visible at the bottom.

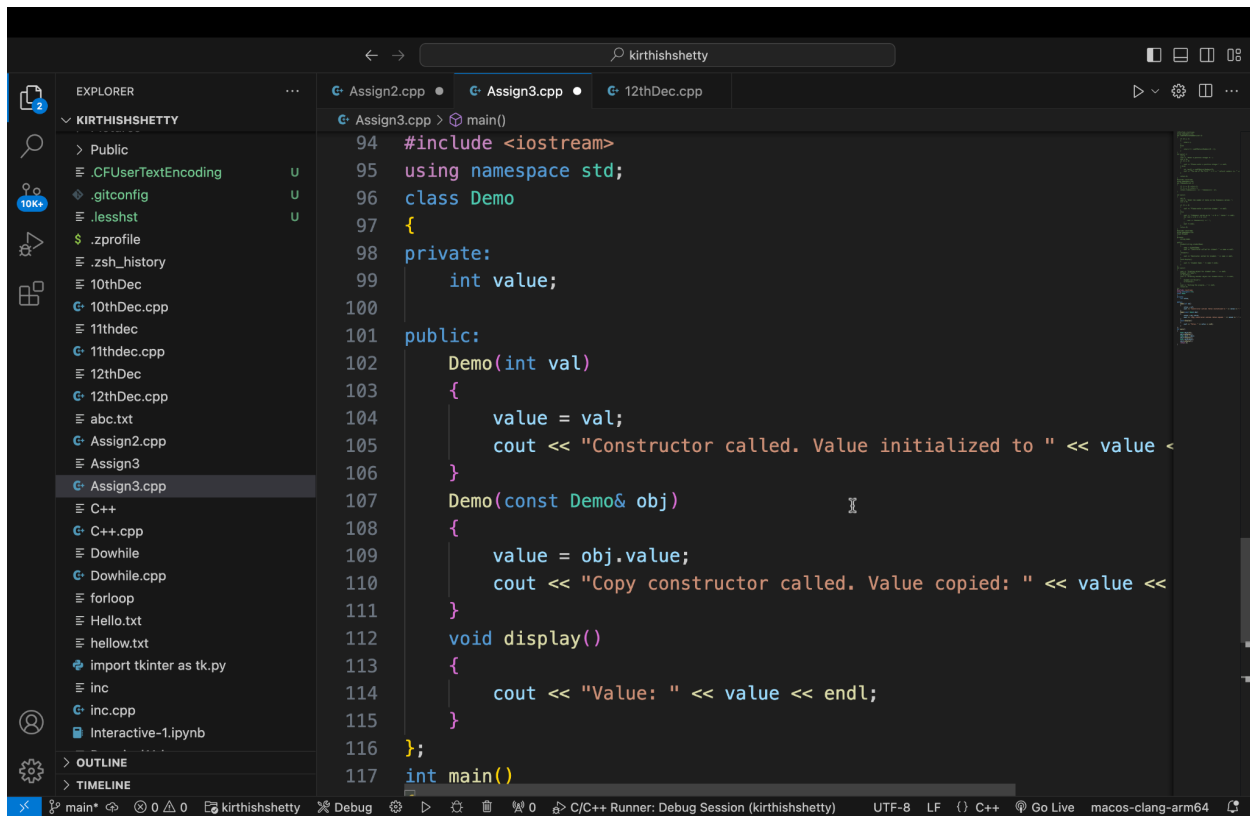
```
1  /*#include <iostream>
59 #include <iostream>
60 using namespace std;
61 class Student
62 {
63 private:
64     string name;
65
66 public:
67     Student(string studentName)
68     {
69         name = studentName;
70         cout << "Constructor called for student: " << name << endl;
71     }
72     ~Student()
73     {
74         cout << "Destructor called for student: " << name << endl;
75     }
76     void display()
77     {
78         cout << "Student Name: " << name << endl;
79     }
80 };
81 int main()
```



This screenshot shows the same Visual Studio Code editor with the 'main()' function of the program. The code creates two 'Student' objects: 's1' with the name 'John' and 's2' with the name 'Alice'. It calls the 'display()' method for each object. After the objects go out of scope, the destructor is called for each, as indicated by the output in the console (not shown here). The program ends by printing 'Exiting the program...' and returning 0.

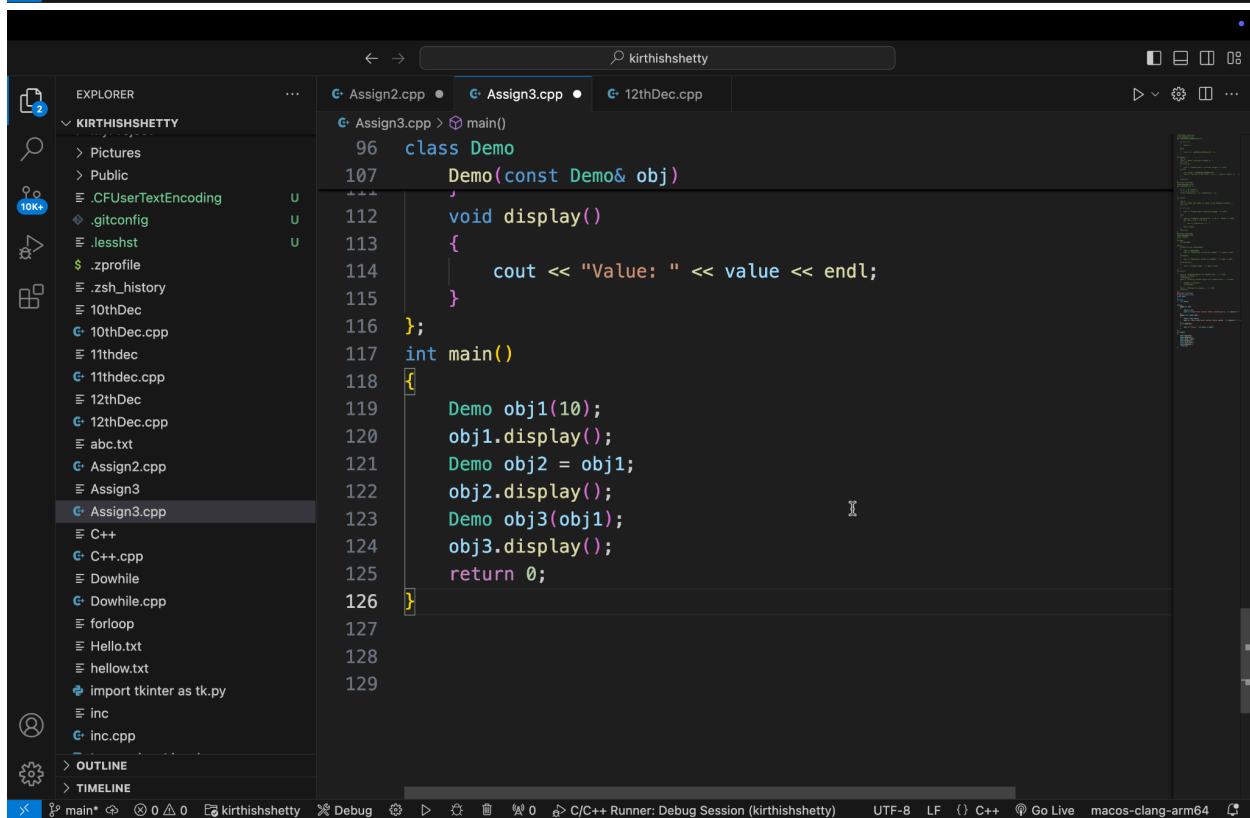
```
61 class Student
76     void display()
78     {
79         cout << "Student Name: " << name << endl;
80     }
81 };
82 int main()
83 {
84     cout << "Creating object for student John..." << endl;
85     Student s1("John");
86     s1.display();
87     cout << "Creating another object for student Alice..." << endl;
88     {
89         Student s2("Alice");
90         s2.display();
91     }
92     cout << "Exiting the program..." << endl;
93     return 0;
94 }
95
```

4.WAP to demonstrate a copy constructor in C++



This screenshot shows the Visual Studio Code editor with the file 'Assign3.cpp' open. The code defines a class 'Demo' with a private member 'int value;'. It includes a constructor 'Demo(int val)' that initializes 'value' and prints a message. It also includes a copy constructor 'Demo(const Demo& obj)' that copies the 'value' from the object and prints a message. A 'display()' method is also defined to print the current value. The 'main()' function is partially visible at the bottom.

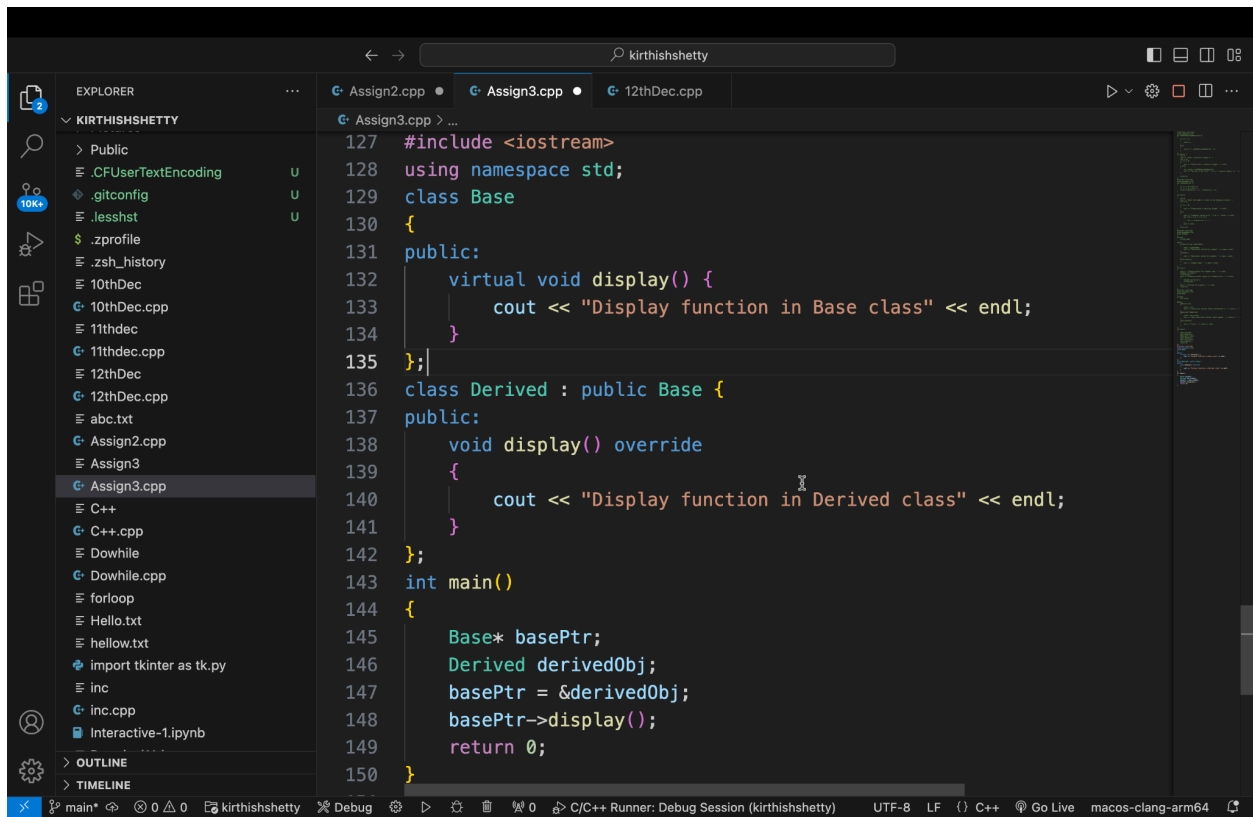
```
94 #include <iostream>
95 using namespace std;
96 class Demo
97 {
98 private:
99     int value;
100
101 public:
102     Demo(int val)
103     {
104         value = val;
105         cout << "Constructor called. Value initialized to " << value << endl;
106     }
107     Demo(const Demo& obj)
108     {
109         value = obj.value;
110         cout << "Copy constructor called. Value copied: " << value << endl;
111     }
112     void display()
113     {
114         cout << "Value: " << value << endl;
115     }
116 };
117 int main()
```



This screenshot shows the same Visual Studio Code editor with 'Assign3.cpp', but the view is scrolled down to the 'main()' function. The code demonstrates the use of the 'Demo' class by creating an object 'obj1' with a value of 10, displaying it, creating a second object 'obj2' as a copy of 'obj1', displaying it, and then creating a third object 'obj3' as a copy of 'obj1', displaying it, and finally returning 0.

```
118 {
119     Demo obj1(10);
120     obj1.display();
121     Demo obj2 = obj1;
122     obj2.display();
123     Demo obj3(obj1);
124     obj3.display();
125     return 0;
126 }
```

5.WAP to demonstrate virtual function in C++



```
127 #include <iostream>
128 using namespace std;
129 class Base
130 {
131 public:
132     virtual void display() {
133         cout << "Display function in Base class" << endl;
134     }
135 };
136 class Derived : public Base {
137 public:
138     void display() override
139     {
140         cout << "Display function in Derived class" << endl;
141     }
142 };
143 int main()
144 {
145     Base* basePtr;
146     Derived derivedObj;
147     basePtr = &derivedObj;
148     basePtr->display();
149     return 0;
150 }
```