Theory Assigment II.

1) Differentiate between function overriding and function overloading. explain with suitable example.

=) Difference between function overriding and function overloading are.

1) function overloading happens in the same class when we declare same function with different arguments in the same class. function overriding is happens in the child class when child class overrides parent class function.

2) In function overloading function signature should be different for all the overloaded functions. In function overriding the signature of both the functions should be same.

3.) overloading happens at the compile time that's why it is also called as compile time polymorphism which overriding happens at run time which is known as run time polymorphism.

4) In function overloading we can have any number of over loaded functions. In function overriding we can have only one overriding function in the child class.

overriding example.

```
class Dog {
  public: void bark() {
    cout<<"woof";
  }
};
```

```
class Hound: public Dog {
    public:
        void bark()
        { cout<< "sniff";
        }
int main()
    {
        Dog D1;
        Hound H;
        H1.bark()
        D1.bark();
```

Overloading example:

```
class Dog{
public
    void bark()
        { cout<< "woof"; }
    void bark(int num) {
    for (int i=0; i<num; i++)
        { cout<< "woof");
        }
    }.
```

2. What does this pointer point to? what are the application of this pointer? Explain with an example.

⇒ This keyword is used to represent an object that invokes the member function. It points to the object for which this function was called. It is automatically passed to a member function when it is called.

```
class myclass
{int i;
 float f;
 pubic:
 myclass()
 { i=0
   f=0.0;
 }
 myclass(int x, float y)
 { i=x;
   f=y;
 }
myclass operator ++()
 { i=i+1;
   f=f+1.0
   return *this // this pointer which
                   points to the called object.
 }
myclass show()
 { cout<<"The elements are"<<i<<f; //accessing dat
                                     member using th
};
```

```
int main()
 { myclass q;
   ++q;
   q.show();
   return 0;
 }
```

output

The elements are.

1

1.0

3. What do you mean by function Template and class Template? Illustrate with examples.

=) function templates are special functions that can operate with generic type. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

Likewise function templates, we ~~might have~~ can write the class to all different data types.

4) Define an a abstract class. Differentiate between virtual and pure virtual functions.

=) An Abstract class is a class that has at least one pure virtual function (i.e. a function that has no definition.).

virtual function	pure virtual function.
1) A virtual function is a member function of base class which can be redefined by derived class.	1) A pure virtual function is a member function of base class whose only declaration is provided in base class and should be defined in derived class otherwise derived class also becomes abstract.
2) classes having virtual functions are not abstract.	2) Base class containing pure virtual function becomes abstract
3) Base class have virtual function can be instantiated. ie. ih object can be made.	4) Base class having pure virtual function becomes abstract ie. it cannot be instantiated.

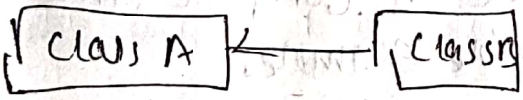
5. What is inheritance? Why is it important? Explain different modes of inheritance in c++ with examples.

⇒ Inheritance is a mechanism where you can derive a class from another class for a hierarchy of classes that share a set of attributes and methods.

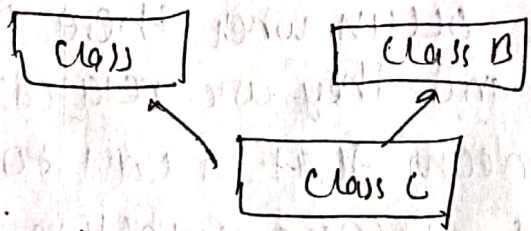
The main advantages of inheritance are code reusability and readability. When child class inherits the properties and functionality of parent class. we need not to write the same code again in child class. This makes it easier to reuse the code.

Types of inheritance are

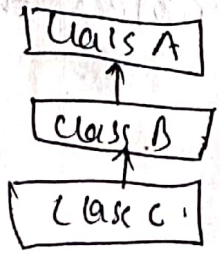
i) Single inheritance



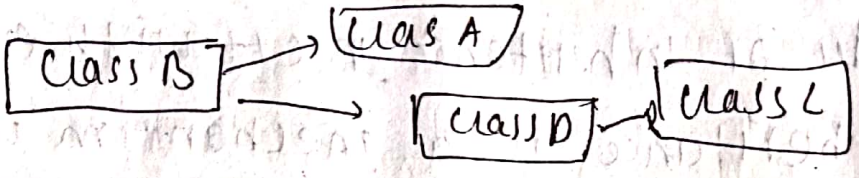
ii) Multiple inheritance



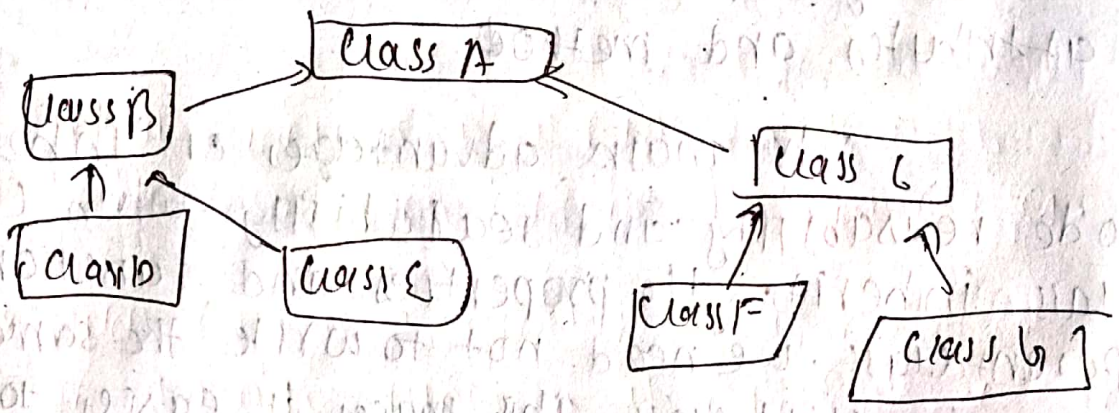
iii) Multilevel inheritance



iv) Hybrid Inheritance



v) Hierarchical Inheritance



c) what does polymorphism mean in c++ language. How is polymorphism achieved in compile time and run time? Explain with suitable example.

=) The word polymorphism means having many forms. polymorphism occur when there is a hierarchy of classes and they are related by inheritance. c++ polymorphism means that a call to a function will cause a different function to be executed depending on the type of object that invokes the function.

i) Compile time polymorphism is achieved by function overloading or operator overloading.

ii) Runtime polymorphism is achieved by function overriding.

7. How do you achieve reusability in C++? what are the advantages of reusability? Class D is derived from class B. Class D doesn't contain any data member on its own. Does class D require constructors? If yes, illustrate with an example.

=> We achieve code reusability in C++ by using object and classes. This help to organize our code and also helps to not write same code for same type in different places.

8. What sort of ambiguity can be resolved by defining virtual base classes? Differentiate between a virtual base class and virtual function.

=> ~~This helps~~ If we didn't specify the inheritance as virtual, the compiler would try to add two Base classes in the derived object, and if

wouldn't know which data to fill in. you would effectively have two objects that are the same in your derived class. Declaring the inheritance as virtual tells the compiler that you want only one instance of the base class in the derived class.

A virtual function is a function that is expected to be overridden by the derived classes.

A virtual class is a similar idea, however instead of a single function it represents a sub part of some ~~funct~~ other object. So, it's a nested class that is derived inside some other class, and has a standard set of virtual functions that each of the derived classes should reimplement for their own particular needs.

9. what is generic programming? How is it implemented in C++? Explain with an example.

=>

Generic is the idea to allow type (integer, string.. etc and user-defined types) to be a parameter to methods, classes and interface. ~~for exa~~

Generic can be implemented in C++ using Templates. Template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data type.

10. Explain how composition differs from Inheritance? Some programmers prefer not to use protected access because they believe it breaks the encapsulation of the base class. Discuss the relative merits of using protected access vs using private access in base classes. In what cases a protected access specifier is generally used?

=>

Inheritance	composition
i) It is the functionality by which one object acquires the characteristics of one or more other objects.	i) Using an object within another object is known as composition.
ii) Class inheritance is defined at ~~class~~ compile-time.	ii) Object composition is defined dynamically at run time.
iii) No access control in inheritance.	iii) Access can be restricted in composition
Exposes both public and protected members of the base class.	The internal details are not exposed to each other and they interact through their public interface.

Protected inheritance helps to gain the structure of your inheritable classes without sharing the class structure only sharing with children and friends.

Method to use protected are it allow faster accessing to base classes. protected field and method, limits implementation changs to a class and its children. limits functions and classes outside of a class's children and friends from altering and accessing data to a class's public interface.

11. Define template. Differentiable function overloading & template overloading with examples

⇒ A template is a blue print or formula for creating a generic class or a function.

function overloading deals with same function name with different arguments but for the same same function in template overloading defining using template helps for every data type, not only for a single type.

Theory program

1-1:

//overriding function

```
#include <iostream>
using namespace std;
class Dog
{

public:
   void bark()
   {
      cout << "Woff" <<endl;
   }
};
class Hound : public Dog
{

public:
   void bark()
   {
      cout << "sniff"<<endl;
   }
};

int main()
{
   Dog D1;
   Hound H1;
   H1.bark();
   D1.bark();
}
```

1-2.

//Function Oveloading

```
#include<iostream>
using namespace std;
class Dog{
   public:
   void bark(){
      cout<<"Woff"<<endl;
```

```
    };
    void bark(int num){
        for(int i=0;i<num;i++){
            cout<<"Wooooooooof";
        }
    }

};
int main(){
    Dog d1;
    d1.bark();
    d1.bark(3);
}
```

2.

```
#include<iostream>
using namespace std;

class Myclass{

    int i;
    float f;
    public:
    Myclass()
{
    i=0;
    f=0.0;
}
Myclass(int x,float y){
    i=x;
    f=y;
}

Myclass operator ++(){
    i=i+1;
    f=f+1.0;
    return *this ;//this pointer which points to the called object
}

Myclass Show(){
    cout<<"The Elment are "<<i <<"\n"<<f<<endl;

}
```

```
};
int main(){
   Myclass a;
   ++a;
   a.Show();
   return 0;

}
```

3-1:

```
#include <iostream>
using namespace std;

template <typename T>
void func_swap(T &arg1, T &arg2)
{
  T temp;
  temp = arg1;
  arg1 = arg2;
  arg2 = temp;
}

int main()
{
  int num1 = 10, num2 = 20;
  double d1 = 100.53, d2 = 435.54;
  char ch1 = 'A', ch2 = 'Z';

  cout << "Original data\n";
  cout << "num1 = " << num1 << "\tnum2 = " << num2<<endl;
  cout << "d1 = " << d1 << "\td2 = " << d2<<endl;
  cout << "ch1 = " << ch1 << "\t\tch2 = " << ch2<<endl;

  func_swap(num1, num2);
  func_swap(d1, d2);
  func_swap(ch1, ch2);

  cout << "\n\nData after swapping\n";
  cout << "num1 = " << num1 << "\tnum2 = " << num2<<endl;
  cout << "d1 = " << d1 << "\td2 = " << d2<<endl;
  cout << "ch1 = " << ch1 << "\t\tch2 = " << ch2<<endl;

  return 0;
}
```

3-2:

```
#include <iostream>
using namespace std;

template <class T>
class myclass {
  T a, b;
 public:
  myclass (T first, T second)
    {a=first; b=second;}
  T getMaxval ();
};
template <class T>
T myclass<T>::getMaxval ()
{
  return (a>b? a : b);
}
int main () {
  myclass <int> myobject (100, 75);
  cout<<"Maximum of 100 and 75 = "<<myobject.getMaxval()<<endl;

  myclass<char> mychobject('A','a');
  cout<<"Maximum of 'A' and 'a' = "<<mychobject.getMaxval()<<endl;

  return 0;
}
```

6-1:

```
#include<iostream>
using namespace std;

class Complex {
private:
    int real, imag;
public:
    Complex(int r = 0, int i =0)  {real = r;   imag = i;}

    // This is automatically called when '+' is used with
    // between two Complex objects
    Complex operator + (Complex const &obj) {
         Complex res;
         res.real = real + obj.real;
         res.imag = imag + obj.imag;
         return res;
```

```
    }
    void print() { cout << real << " + i" << imag << endl; }
};

int main()
{
    Complex c1(10, 5), c2(2, 4);
    Complex c3 = c1 + c2; // An example call to "operator+"
    c3.print();
}
```

6-2:

```
#include <bits/stdc++.h>
using namespace std;

class base
{
public:
    virtual void print ()
    { cout<< "print base class" <<endl; }

    void show ()
    { cout<< "show base class" <<endl; }
};

class derived:public base
{
public:
    void print () //print () is already virtual function in derived class, we could also declared as virtual
void print () explicitly
    { cout<< "print derived class" <<endl; }

    void show ()
    { cout<< "show derived class" <<endl; }
};

//main function
int main()
{
    base *bptr;
    derived d;
    bptr = &d;

    //virtual function, binded at runtime (Runtime polymorphism)
    bptr->print();

    // Non-virtual function, binded at compile time
    bptr->show();
```

```
    return 0;
}
```

7.

```
//class D doesn't requiere the constructor as it inherits everything from base class;

#include <iostream>
using namespace std;

class B
{

public:
    int x;
    B()
    {
        x = 0;
    }
    B(int z)
    {
        x = z;
    }
    void display()
    {
        cout << "The Number is " << x << endl;
    }
};

class D : public B
{
};

int main()
{
    D a;
    a.x = 3;
    a.display();
}
```

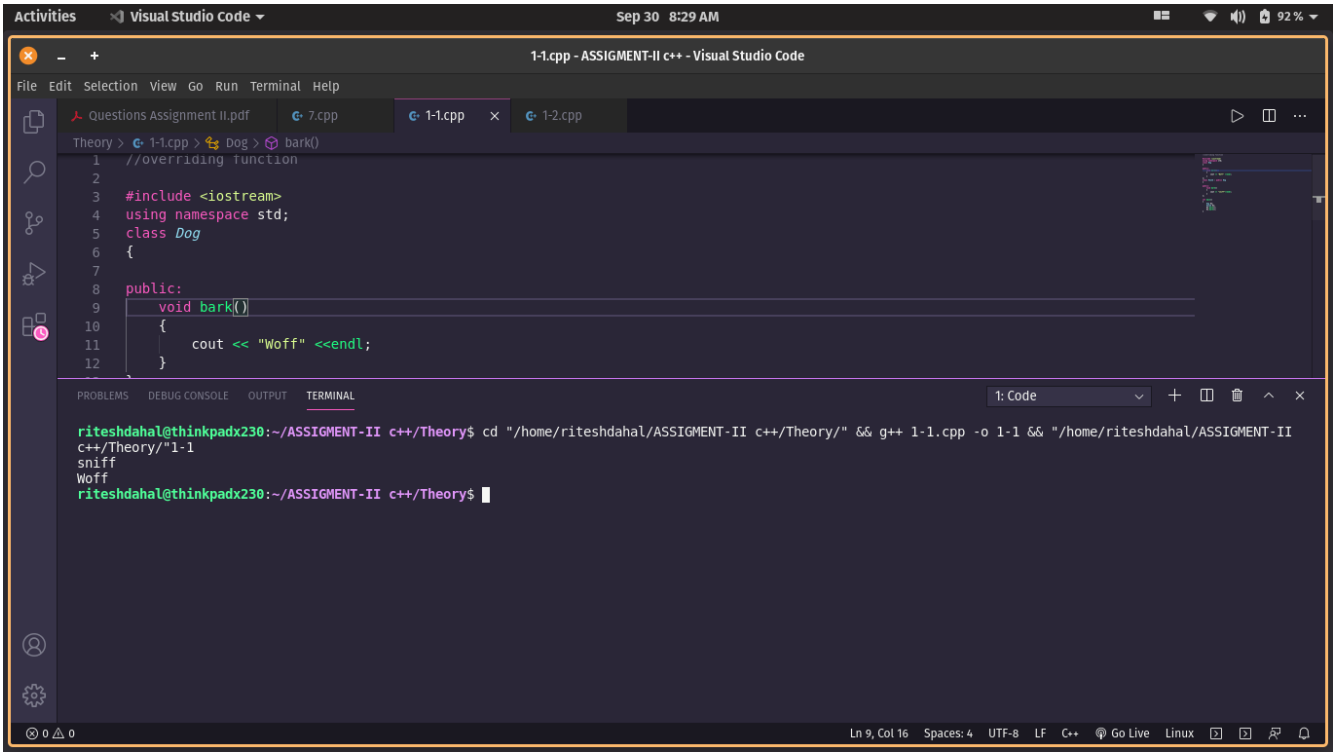
9.

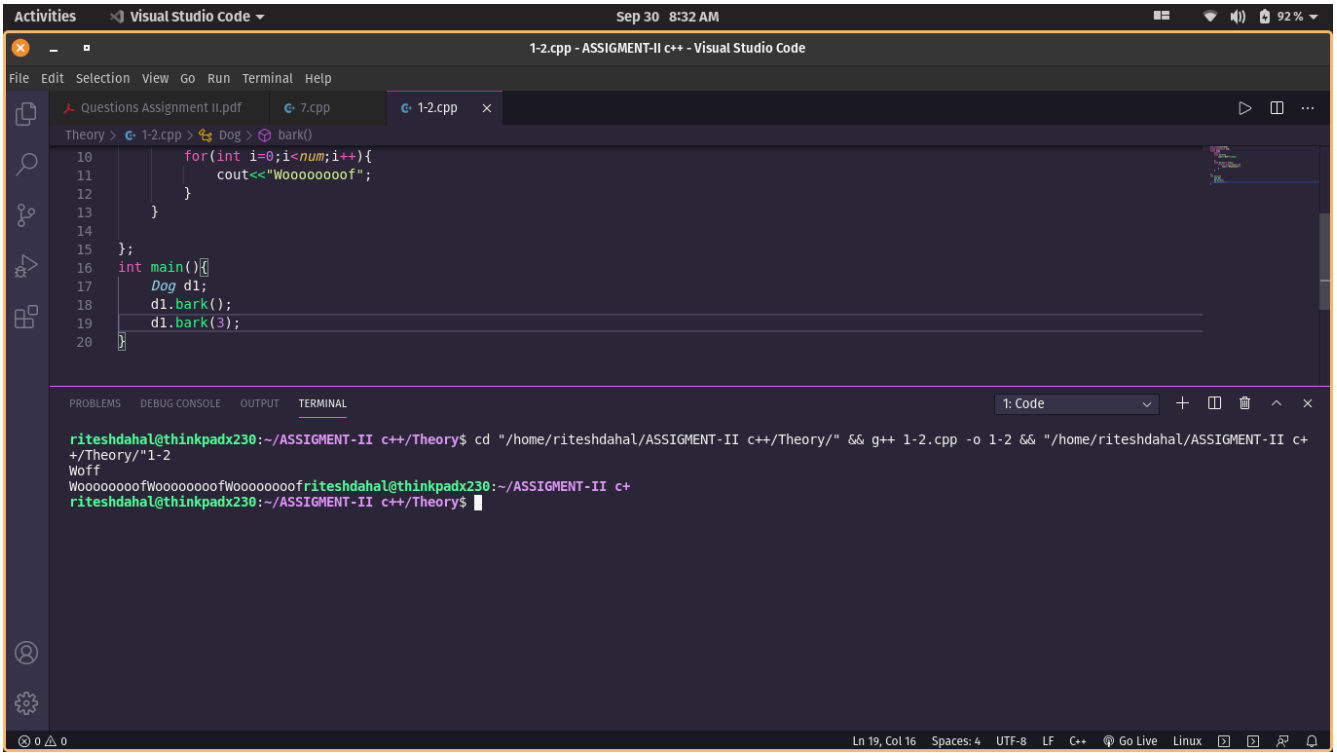
```
#include <iostream>
using namespace std;
```

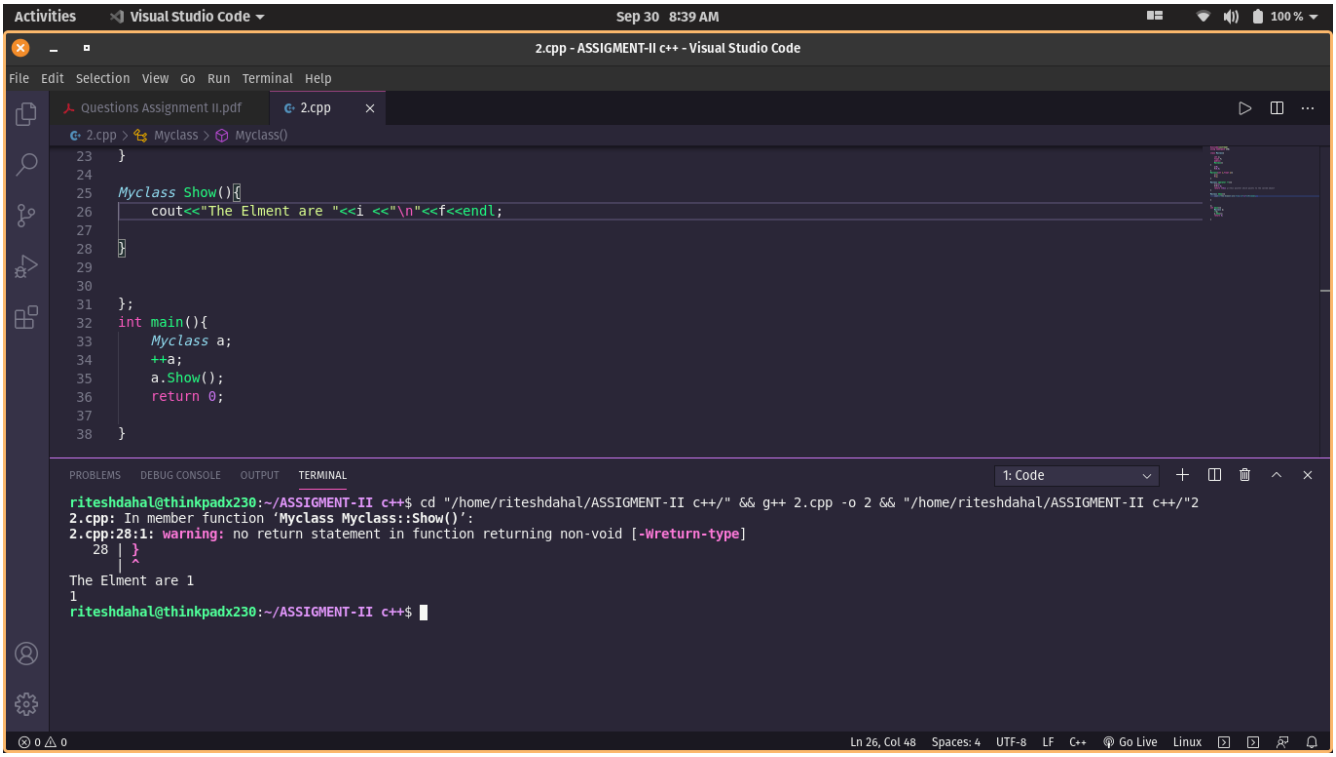
```
// One function works for all data types.
// This would work even for user defined types
// if operator '>' is overloaded
template <typename T>

T myMax(T x, T y)
{
   return (x > y) ? x : y;
}

int main()
{

   // Call myMax for int
   cout << myMax<int>(3, 7) << endl;

   // call myMax for double
   cout << myMax<double>(3.0, 7.0) << endl;

   // call myMax for char
   cout << myMax<char>('g', 'e') << endl;

   return 0;
}
```

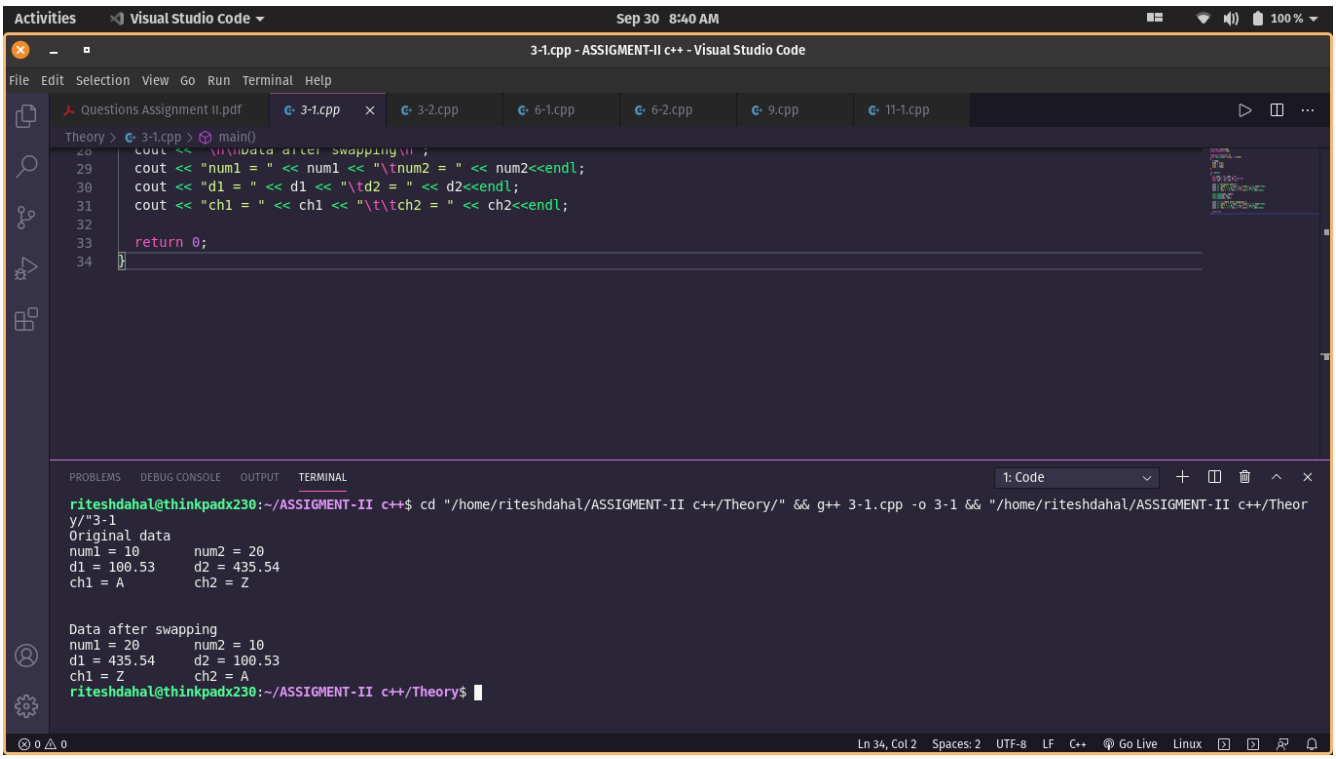
11.

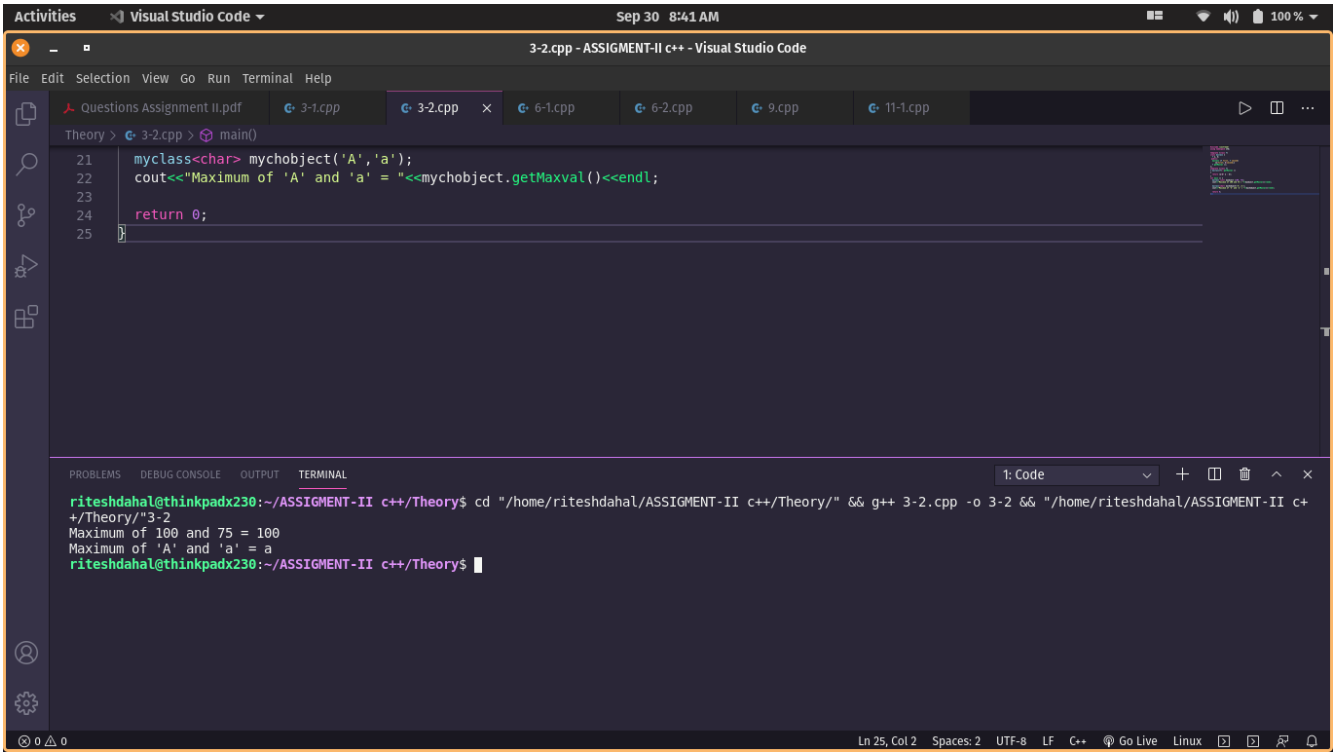
```
#include <iostream>
using namespace std;

// One function works for all data types.  This would work
// even for user defined types if operator '>' is overloaded
template <typename T>
T myMax(T x, T y)
{
  return (x > y)? x: y;
}

int main()
{
 cout << myMax<int>(3, 7) << endl;  // Call myMax for int
 cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double
 cout << myMax<char>('g', 'e') << endl;   // call myMax for char

 return 0;
}
```

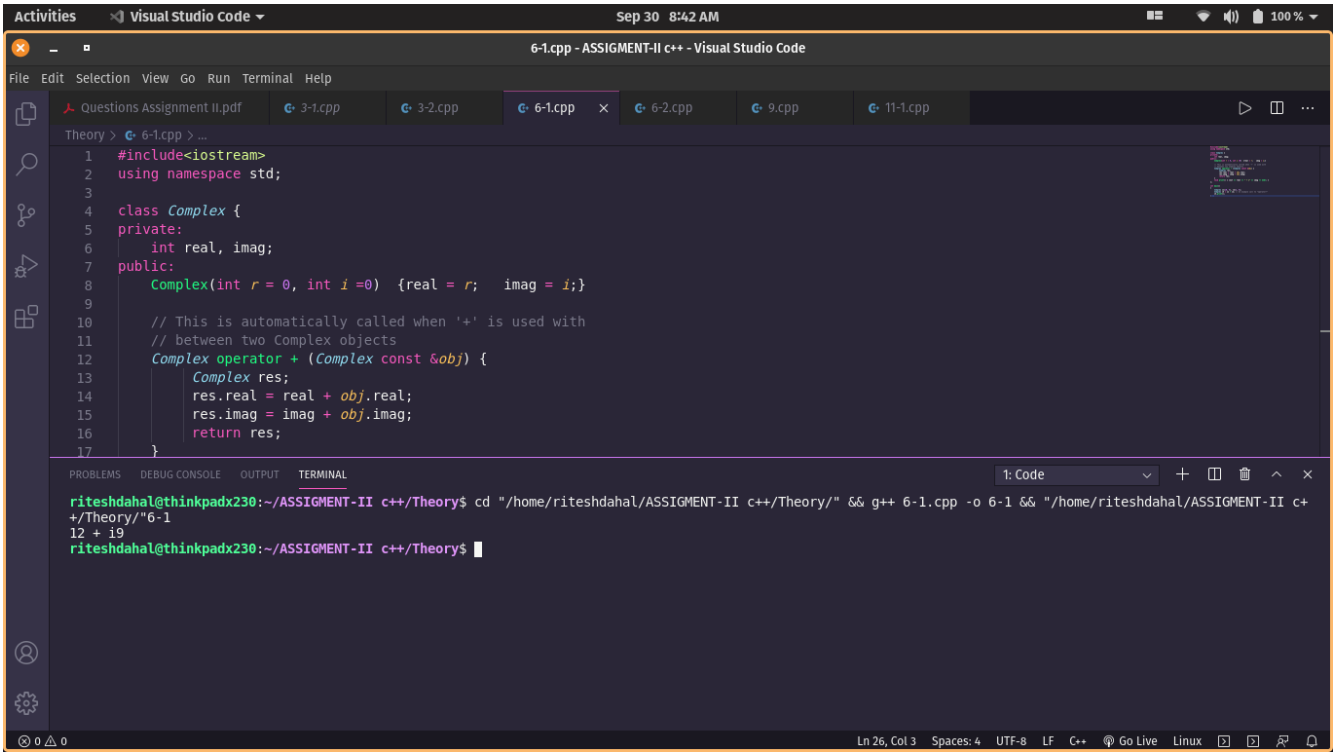

Activities
Visual Studio Code
Sep 30 8:29 AM
92 %
1-1.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
7.cpp
1-1.cpp
1-2.cpp
Theory > 1-1.cpp > Dog > bark()
//overriding function
#include <iostream>
using namespace std;
class Dog
{
public:
void bark()
{
cout << "Woff" <<endl;
}
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 1-1.cpp -o 1-1 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"1-1
sniff
Woff
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 9, Col 16 Spaces: 4 UTF-8 LF C++ Go Live Linux

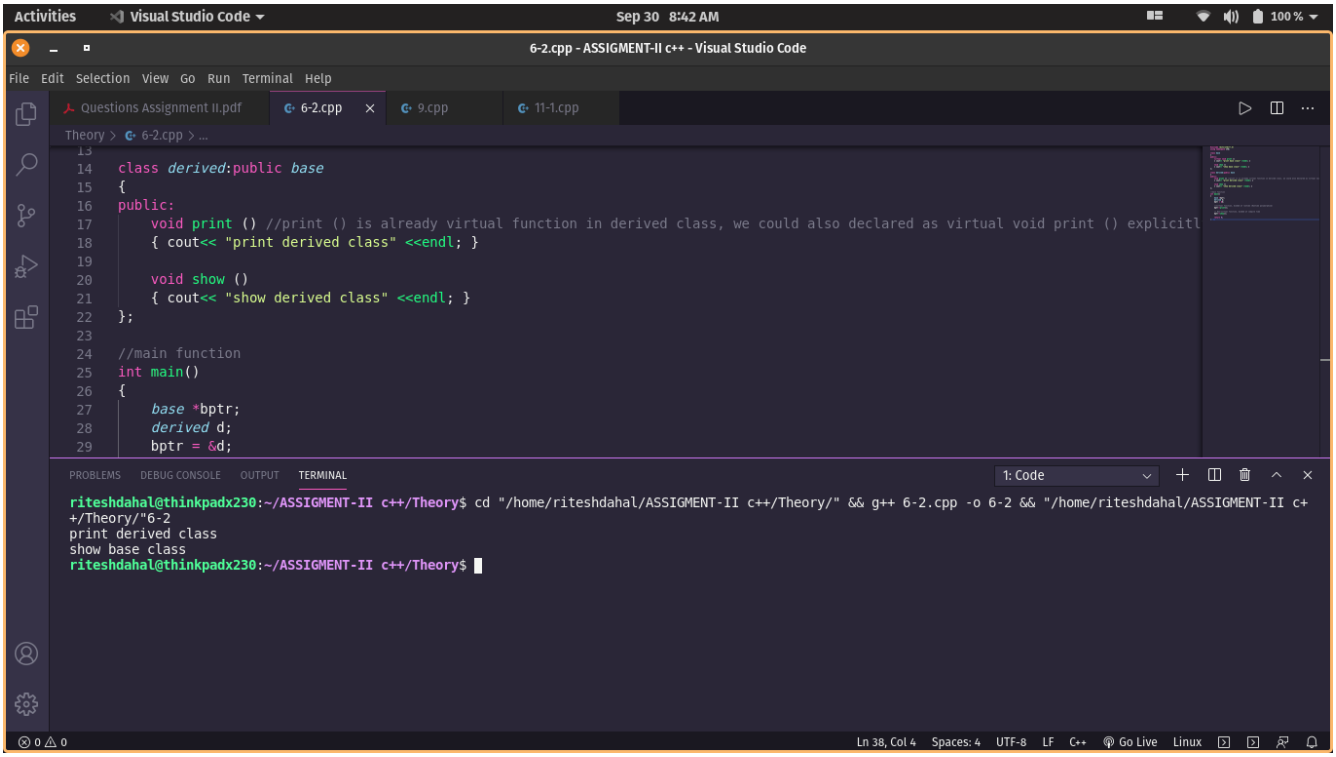

Activities
Visual Studio Code
Sep 30 8:32 AM
92 %
1-2.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
7.cpp
1-2.cpp
Theory > 1-2.cpp > Dog > bark()
for(int i=0;i<num;i++){
cout<<"Wooooooof";
}
}
};
int main(){
Dog d1;
d1.bark();
d1.bark(3);
}
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 1-2.cpp -o 1-2 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"1-2
Woff
WooooooofWooooooofWooooooofriteshdahal@thinkpadx230:~/ASSIGMENT-II c+
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 19, Col 16 Spaces: 4 UTF-8 LF C++ Go Live Linux

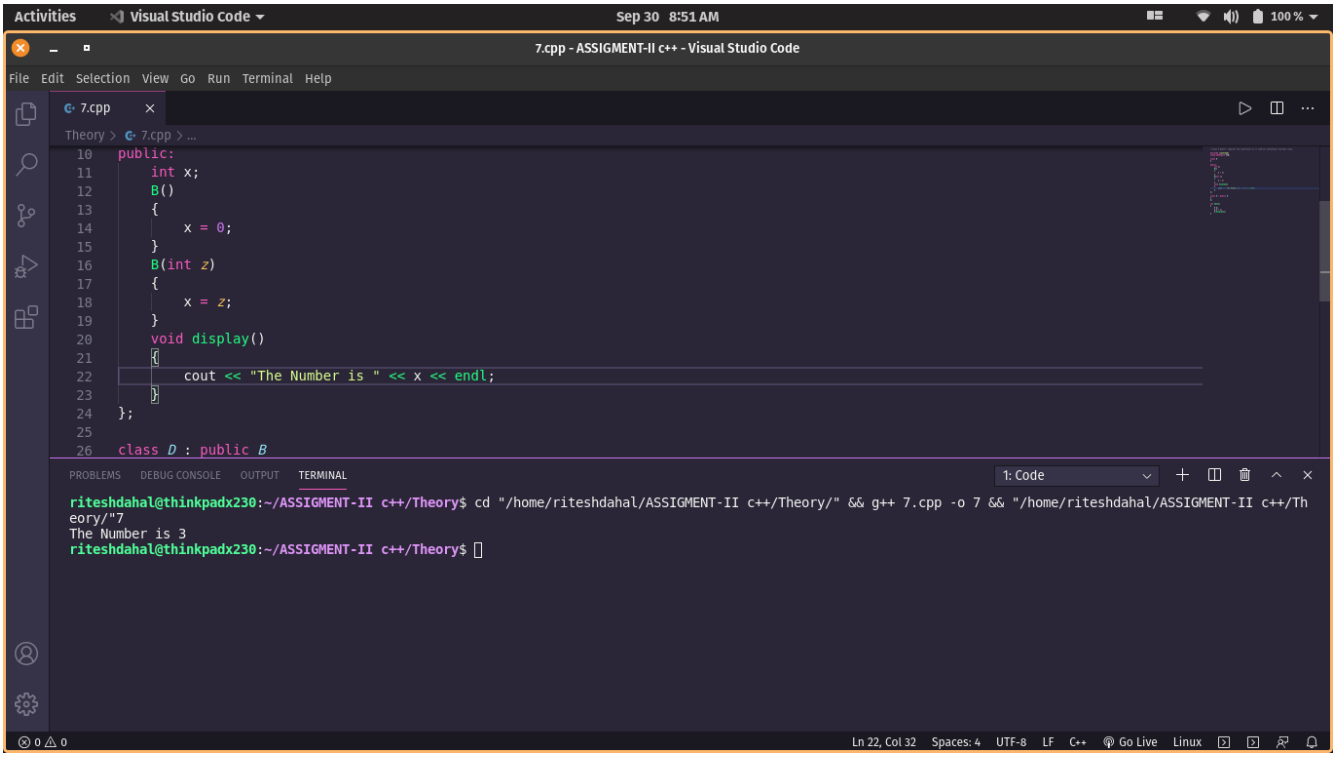

Activities
Visual Studio Code
Sep 30 8:39 AM
2.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
2.cpp
2.cpp > Myclass > Myclass()
23 }
24
25 Myclass Show(){
26 cout<<"The Elment are "<<i <<"\n"<<f<<endl;
27
28 }
29
30
31 };
32 int main(){
33 Myclass a;
34 ++a;
35 a.Show();
36 return 0;
37
38 }
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++$ cd "/home/riteshdahal/ASSIGMENT-II c++/" && g++ 2.cpp -o 2 && "/home/riteshdahal/ASSIGMENT-II c++/"2
2.cpp: In member function 'Myclass Myclass::Show()':
2.cpp:28:1: warning: no return statement in function returning non-void [-Wreturn-type]
28 | }
| ^
The Elment are 1
1
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++$
0 0
Ln 26, Col 48 Spaces: 4 UTF-8 LF C++ Go Live Linux

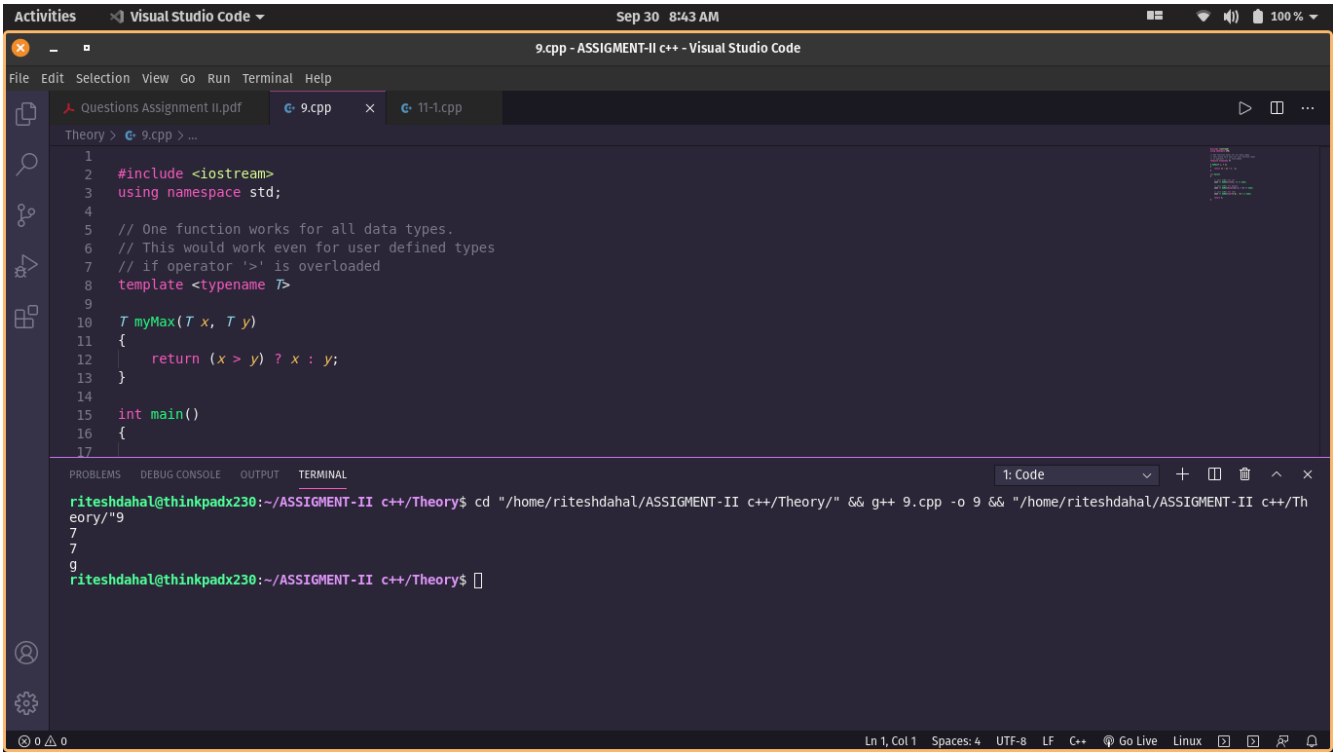

Activities
Visual Studio Code
Sep 30 8:40 AM
3-1.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
3-1.cpp
3-2.cpp
6-1.cpp
6-2.cpp
9.cpp
11-1.cpp
Theory > 3-1.cpp > main()
29 cout << "num1 = " << num1 << "\tnum2 = " << num2<<endl;
30 cout << "d1 = " << d1 << "\td2 = " << d2<<endl;
31 cout << "ch1 = " << ch1 << "\t\tch2 = " << ch2<<endl;
32
33 return 0;
34 }
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 3-1.cpp -o 3-1 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"3-1
Original data
num1 = 10 num2 = 20
d1 = 100.53 d2 = 435.54
ch1 = A ch2 = Z
Data after swapping
num1 = 20 num2 = 10
d1 = 435.54 d2 = 100.53
ch1 = Z ch2 = A
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
0 0
Ln 34, Col 2 Spaces: 2 UTF-8 LF C++ Go Live Linux

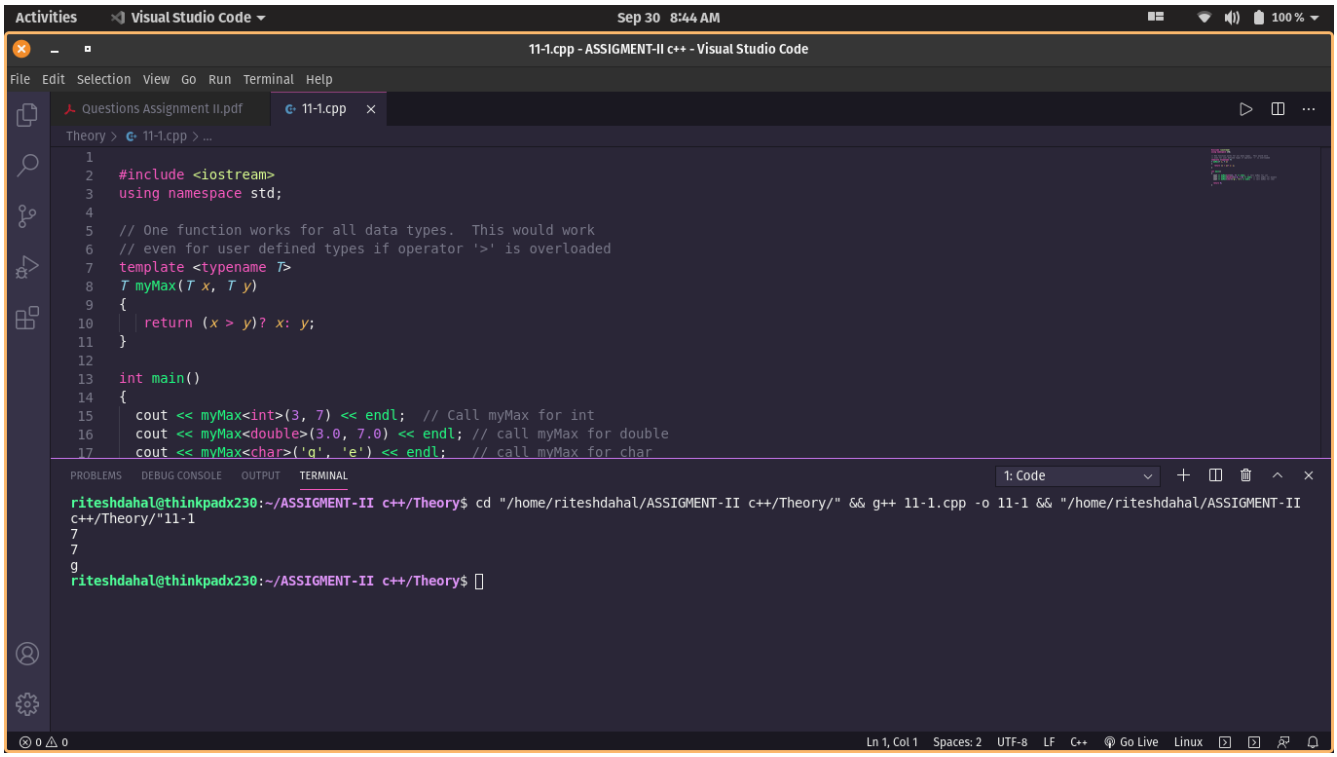

Activities
Visual Studio Code
Sep 30 8:41 AM
3-2.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
3-1.cpp
3-2.cpp
6-1.cpp
6-2.cpp
9.cpp
11-1.cpp
Theory > 3-2.cpp > main()
21 myclass<char> mychobject('A','a');
22 cout<<"Maximum of 'A' and 'a' = "<<mychobject.getMaxval()<<endl;
23
24 return 0;
25 }
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 3-2.cpp -o 3-2 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"3-2
Maximum of 100 and 75 = 100
Maximum of 'A' and 'a' = a
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 25, Col 2 Spaces: 2 UTF-8 LF C++ Go Live Linux


Activities
Visual Studio Code
Sep 30 8:42 AM
6-1.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
3-1.cpp
3-2.cpp
6-1.cpp
6-2.cpp
9.cpp
11-1.cpp
Theory > 6-1.cpp > ...
#include<iostream>
using namespace std;
class Complex {
private:
int real, imag;
public:
Complex(int r = 0, int i =0) {real = r; imag = i;}
// This is automatically called when '+' is used with
// between two Complex objects
Complex operator + (Complex const &obj) {
Complex res;
res.real = real + obj.real;
res.imag = imag + obj.imag;
return res;
}
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 6-1.cpp -o 6-1 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"6-1
12 + i9
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 26, Col 3 Spaces: 4 UTF-8 LF C++ Go Live Linux


Activities
Visual Studio Code
Sep 30 8:42 AM
6-2.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
6-2.cpp
9.cpp
11-1.cpp
Theory > 6-2.cpp > ...
class derived:public base
{
public:
void print () //print () is already virtual function in derived class, we could also declared as virtual void print () explicitl
{ cout<< "print derived class" <<endl; }
void show ()
{ cout<< "show derived class" <<endl; }
};
//main function
int main()
{
base *bptr;
derived d;
bptr = &d;
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 6-2.cpp -o 6-2 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"6-2
print derived class
show base class
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 38, Col 4 Spaces: 4 UTF-8 LF C++ Go Live Linux


Activities
Visual Studio Code
Sep 30 8:51 AM
7.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
7.cpp
Theory > 7.cpp > ...
public:
int x;
B()
{
x = 0;
}
B(int z)
{
x = z;
}
void display()
{
cout << "The Number is " << x << endl;
}
};
class D : public B
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 7.cpp -o 7 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"7
The Number is 3
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 22, Col 32 Spaces: 4 UTF-8 LF C++ Go Live Linux


Activities
Visual Studio Code
Sep 30 8:43 AM
100 %
9.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
9.cpp
11-1.cpp
Theory > 9.cpp > ...
#include <iostream>
using namespace std;
// One function works for all data types.
// This would work even for user defined types
// if operator '>' is overloaded
template <typename T>
T myMax(T x, T y)
{
return (x > y) ? x : y;
}
int main()
{
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 9.cpp -o 9 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"9
7
7
g
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 1, Col 1 Spaces: 4 UTF-8 LF C++ Go Live Linux


Activities
Visual Studio Code
Sep 30 8:44 AM
100 %
11-1.cpp - ASSIGMENT-II c++ - Visual Studio Code
File Edit Selection View Go Run Terminal Help
Questions Assignment II.pdf
11-1.cpp
Theory > 11-1.cpp > ...
#include <iostream>
using namespace std;
// One function works for all data types. This would work
// even for user defined types if operator '>' is overloaded
template <typename T>
T myMax(T x, T y)
{
return (x > y)? x: y;
}
int main()
{
cout << myMax<int>(3, 7) << endl; // Call myMax for int
cout << myMax<double>(3.0, 7.0) << endl; // call myMax for double
cout << myMax<char>('g', 'e') << endl; // call myMax for char
PROBLEMS DEBUG CONSOLE OUTPUT TERMINAL
1: Code
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$ cd "/home/riteshdahal/ASSIGMENT-II c++/Theory/" && g++ 11-1.cpp -o 11-1 && "/home/riteshdahal/ASSIGMENT-II c++/Theory/"11-1
7
7
g
riteshdahal@thinkpadx230:~/ASSIGMENT-II c++/Theory$
Ln 1, Col 1 Spaces: 2 UTF-8 LF C++ Go Live Linux