Hands-on:-

Part-1: Classes and Objects
- Box
- Account
- Point
- Color
- Image
- IPAddress
- MyTime
- MyDate
- MyStack
- MyString

Part-2 : Operator Overloading
- Complex
- MyTime
- Fraction class
- Currency / Weight / Distance
- MyDate
- Matrix
- MyString

Part-3 : Inheritance & Virtual Functions
- Geometric Shapes
- Banking Accounts
- Mobile Billing Customers

Guidelines/Expectations
* Class Diagram
* Multifile/modular coding

* member wise initializer list for simple members
* copy ctor, dtor only if needed (Non Trivial)
* const suffix for immutable operations

* Test cases using googletest
* Prefer use std:: prefix (std::cout, std::cin)
  avoid "using namespace std;"

* Getter/Accessor functions
* Avoid setter functions as much possible

* meaningful names
* naming conventions
* code style
  (indentation, formatting)
----
* static analysis
* heap analysis
  /memory leak detection

```
Point
+ quadrant()
+ distanceFromOrigin()
+ getter functions

Color
- m_red
- m_green
- m_blue
+ Color(r:Int,g:Int,b:Int)
+ Color(hexcode:Int)
+ Color(hexstr:String)
+ hexval()
+ invert()
+ display()
+ getter functions

Image
- m_x : Int    // x pos
- m_y : Int    // y pos
- m_width : Int
- m_height : Int
+ rotate()
+ scale()  // zoom()
+ shift()  // move()
+ display()
```

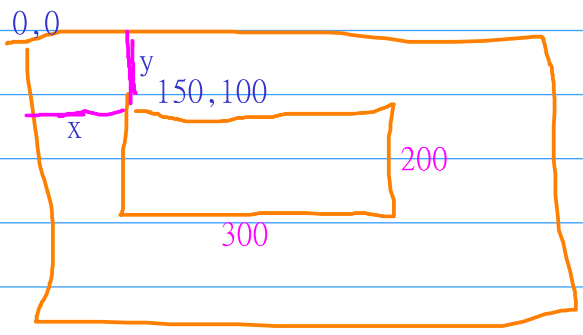
```
IPAddress
- ipval : uint32_t
(or)
- a : uint8_t
- b : uint8_t
- c : uint8_t
- d : uint8_t

+ ipval() : uint32_t
+ ipstr() : string
+ display()
+ isLoopback()
+ class() //A,B,C,D
```

```
Color c2(0xFF00FF)
Color c3("7F2352");

192.168.72.25

IPAddress ip1(192,168,72,25)
IPAddress ip2("192.168.72.25")
IPAddress ip3(0xC0A84819)
IPAddress ip4; //127.0.0.1
```



```
TEST(BoxTest, BoxGetter())
{
   Box b1(10,12,5);
   EXPECT_EQ(10,b1.length())
   EXPECT_EQ(12,b1.breadth())
   EXPECT_EQ(5,b1.height())
}
TEST(BoxTest, BoxVolume)
{
   Box b1(10,12,5);
   EXPECT_EQ(600, b1.findVolume())
}
TEST(BoxTest, BoxSurfaceArea())
{
   Box b1(10,12,5);
   EXPECT_EQ(460,b1.findSurfaceArea())
}
```

Test cases should be atomic
as much as possible..
Let's not combine two operations
in single test case

```
IPAddress ip2("192.0.49.35");
EXPECT_FALSE(ip2.isLoopback())

IPAddress ip2("192.0.49.35");
EXPECT_EQ(0xC0003123. ip2.hexval())

IPAddress ip2("192.0.49.35");
EXPECT_STRCASEEQ("C0003123", ip2.hexstr())

IPAddress ip4("127.0.0.1");
EXPECT_TRUE(ip4.isLoopback())
```

EXPECT_TRUE
EXPECT_FALSE

EXPECT_STREQ
EXPECT_STRNE

EXPECT_EQ
EXPECT_NE
EXPECT_LE
EXPECT_LT
EXPECT_GE
EXPECT_GT

```
class ICustomer         //abstract, like interface in Java
{
   public:
   void makeCall(int)=0; //duration in minutes
   void credit(double)=0; //recharge or bill amount
};
class Customer : public ICustomer       //also abstract
{
    std::string m_id;
    std::string m_name;
    double m_balance;
    public:
    Customer(string id, string name, double balance);
    double balance() { return m_balance;}
    //TODO : display
};
class PrepaidCustomer : public Customer
{
    //if any other data members needed
    public:
    Customer(string id, string name, double balance):Customer(id,name,balance) { }
    void credit(double);
    void makeCall(int);
};
class PostpaidCustomer : public Customer
{
        //similar to Prepaid
};

ICustomer *pcust;
if(cond)
   pcust = new PrepaidCustomer(/*...*/)
else
   pcust = new PostpaidCustomer(/*...*/)

pcust->makeCall(10);
pcust->credit(500);
```

```
 customer.h                customer.h
 customer.cpp              precustomer.h
 main.cpp / test.cpp       postcustomer.h
                           customer.cpp
                           precustomer.cpp
                           postcustomer.cpp
                           main.cpp / test.cpp
```