**Chimera Investment Corporation Test Question**

**Preamble:**

The following bond-yield calculator assignment is designed to help us understand how you think, how you transcribe your logic in code, and the standards to which you try to adhere. You can do the assignment in C, C++, C#, or Java, and you can create either a Windows GUI or a console application. **To give you a standard of quality to strive for, assume that we will copy the binary of your working application to the desktops of ~30 traders, none of whom are programmers, and assume that some traders will use it every day, while others will use it only occasionally. They will use either the GUI, or type in commands**.

**Bond Yield Calculator Background:**

Fixed-income securities, such as a bond, are priced according to the cash flows that the holder will receive, and when they will receive them. Cash received sooner is worth more than cash received later, simply because the holder can reinvest the cash they receive now and realize a gain. If they have to wait for the cash to come in, the holder will not be able to reinvest it until later.

The gain that the holder can realize on a cash flow is based on the current interest rate. For the sake of simplicity, let’s say there is a cash flow CF, and that it’s reinvested at an annual rate of *r*. The cash that the holder will receive in N years is simply:

This is the standard and straightforward compound interest calculation.

With this in mind, we can actually discount a cash flow received in N years back to its present value, given a rate *r*. If we receive a cash flow CF1 in N years, we know that its present value CF is simply:

This equation is simply the first equation, but with both sides divided by ( 1 + *r* )N. When it’s written like this, the rate *r* is often referred to as the *discount* rate.

Now a bond is simply a set of cash flows. If a company issues an annual-pay, 5-year, $1000 bond with a 10% coupon, this means two things:

1. The company will receive some amount of money up front from the bondholder, when they purchase the security.
2. In exchange for that money, the company will make two sets of payments to the bondholder: first, the company will make a coupon payment of $100 (10% of $1000) each year for 5 years; second, at the end of the 5th year, the company will make a principal payment of $1000.

So how much money does the bondholder have to pay to receive those cash flows? Namely, how much is the “some amount of money” mentioned above? That can be determined by the discount technique described above, along with an assumed rate. Assuming a discount rate of 15%, the specific bond described above will have the sum of its discounted cash flows – their present values – as given by:

The first 5 terms in the equation are simply the annual coupon payments, discounted to their present values (for example, the coupon payment that is coming 3 years from now has to be discounted by 1.153, the one that is coming 4 years from now has to be discounted by 1.154, etc.). The last term is the principal payment, also discounted to its present value. The sum of all of those present values, $832.4, is *the price of the bond*. This is how much the bondholder will have to pay, up front, in order to receive those cash flows from the issuer.

Why is the price of the bond less than $1000? Well, in an environment where interest rates are 15%, receiving $100 annually on an investment of $1000 is not worthwhile to the holder. They could simply put their money in some other investment at 15%, and get $50 more a year. So the bond has to be offered at a discount; it has to be offered for less than $1000 in order to entice someone to buy it.

If the bond price is $832.4, that means that it will yield a return of 15% to the holder over the lifetime of the bond – and in fact the full name of that return is called yield to maturity (YTM). **Note that in this particular context, YTM is the same thing as the interest rate, and it’s the same thing as the discount rate. The terms yield, yield to maturity, discount rate, and interest rate can be used interchangeably. They all refer to the rate *r*.**

This bond pricing equation can be generalized. For a particular bond with a coupon payment of C, a time of N years, and a face amount of F, the price of the bond for a given rate *r* is:

**Assignment:**

As mentioned in the preamble, your assignment, if you choose to take it, is to write a bond-yield calculator.

The calculator will take as inputs, the coupon, the time, the face value, and the rate. Given that data, your calculator will return the bond’s price.

In addition, your calculator can also take as inputs, the coupon, the time, the face value, and the price, and return the bond’s yield, i.e. return the rate *r*.

Your calculator must be easy enough or intuitive enough for a technical person to run and use, without looking at the code itself. We ought to be able to enter the data and see the results with a minimum of thought.

**Method Signatures:**

We are looking for (at least) two methods with the following signatures:

**double CalcPrice(double coupon, int years, double face, double rate);**

**double CalcYield(double coupon, int years, double face, double price);**

**Your methods should have an accuracy of 10-7 for all results**. Thus, for our current working example, your program should produce the following results**:**

**CalcPrice(0.10, 5, 1000, 0.15) returns 832.3922451**

**CalcYield(0.10, 5, 1000, 832.4) returns 0.1499974**

For a second example, where the coupon rate is the same as the interest rate / yield, your program should produce:

**CalcPrice(0.15, 5,1000, 0.15) returns 1000.0000000**

**CalcYield(0.10, 5,1000, 1000) returns 0.1000000**

(Interesting factoid: when the coupon rate is the same as the yield, the price of the bond matches its face amount.)

And for a third example, your calculator should be able to handle premium bonds (i.e. bonds that sell for more than their face amount, since their coupon is higher than the prevailing discount rate):

**CalcPrice(0.10, 5, 1000, 0.08) returns 1079.8542007**

**CalcYield(0.10, 5, 1000, 1079.85) returns 0.0800001**

Lastly, make sure that your program methods are consistent: if your price calculation gives you a price PPP given a yield YYY, then your yield calculation, given a price PPP, ought to give you YYY right back, as shown here:

Note how entering a price of 528.8807463 gives us a yield of 19%, and a yield of 19% gives us 528.8807463 right back.

**CalcPrice(0.10, 30, 1000, 0.19) returns 528.8807463**

**CalcYield(0.10, 30, 1000, 528.8807463) returns 0.1900000**

**Bonus Assignment:**

Keep track of, and print out, the amount of time it takes your program to calculate the price and yield. How would you optimize your program to reduce it?

**Submission:**

Please have your project / build files configured in such a way that we can build it from a Windows machine, in a directory c:\applicants\[yourname]. If your name is john smith, we’d like to be able to build everything in a single click from c:\applicants\jsmith.

**Final Notes:**

Feel free to use whatever resources you need to in order to solve this problem, and to help you write and test your program.

When you come in, we will go over the program with you in detail. We’ll be specifically looking to make sure that you completely understand how and why it works the way it does, and that you wrote it in a professional manner.

Just as a heads-up, saying that “I found a formula on the Internet”, without understanding why it works, is not a good enough answer.

Anyhow, thank you very much for reading this, and we hope that you enjoy working on the problem – we sure did. And we’re very much looking forward to seeing your solution.

Sincerely,

The Chimera Development Team