ISyE 6767 Sys-Computation Finance Homework 2 Report

1. Bond Class Design

Implementation:

A `Bond` class was designed to represent a security that makes fixed payments on specific dates based on its expiration date, frequency of payments, and coupon rate. The class definition includes:

- **Private Members**:
  - `expiration_date`: A string representing the bond's expiration date.
  - `frequency`: A double indicating how often payments are made.
  - `coupon_rate`: A double representing the bond's coupon rate.
- **Functions**:
  - **Default Constructor**: Initializes all private members to their default values.
  - **Destructor**: Cleans up resources.
  - **Copy Constructor**: Enables deep copying of bond objects.
  - **Parameterized Constructor**: Initializes a bond using the expiration date, frequency of payments, and coupon rate.
  - **ToString()**: Returns a string description of the bond.

In the main application, a default bond is instantiated and its information is printed to the command line. A new bond, with semi-annual payments, a 7% coupon rate, and an expiration date of November 19th, 2035, is also instantiated and its characteristics are printed using the `ToString()` function.

Test Results:

The `ToString()` function was tested using two test cases:

1. Default bond: Expected output is "Bond(00-00-0000,0.0000,0.0000)".
2. A bond with expiration date "01/01/2030", frequency of 2.0, and coupon rate of 0.05: Expected output is "Bond(01/01/2030,2.0000,0.0500)".

![Include your screenshot here]

2. Bond Pricing Function

Implementation:

A bond pricing function was added to the `Bond` class to determine its price based on various factors. When a bond is issued, its price is typically 100. However, post-issuance, the bond's price fluctuates according to the prevailing interest rate.

The price of a bond is computed as the present value of its cash flows:

[
ext{Price} = \sum_{i=1}^{n} CF_i imes DF_i
]

Where:

- (CF_i) is the cash flow at time (i).
- (DF_i) is the discount factor at time (i).

The bond pricing function takes into account the time to maturity, interest rate, coupon rate, payment frequency, and face value to compute the bond's price.

Test Results:

The bond pricing function was tested for accuracy using two test cases:

1. A bond with 5.0 years to maturity, 3% interest rate, 4% coupon rate, and annual payments: Expected price is 104.366.
2. A bond with 10 years to maturity, 2% interest rate, 3% coupon rate, and semi-annual payments: Expected price is 108.523.

```
sprx7767@Pranav-ROG:/mnt/d/gatech/ISYE 6767 - Design and Implementation of Systems to Support
rm main
sprx7767@Pranav-ROG:/mnt/d/gatech/ISYE 6767 - Design and Implementation of Systems to Support
g++ main.cpp bond.cpp -o main
./main
Test ToString with default constructor: PASSED.
Test ToString with parameterized constructor: PASSED.
Test Price case 1: PASSED.
Test Price case 2: PASSED.
Bond(00-00-0000,0.0000,0.0000)
Bond(11-19-2035,0.5000,0.0700)
Bond(11-19-2035,0.5000,0.0700)
103.145
This is a good Investment with a net earnings of 73.5196and gross earnings of 24.4804
sprx7767@Pranav-ROG:/mnt/d/gatech/ISYE 6767 - Design and Implementation of Systems to Support
```

3. Bond Average Price from 2016 to 2020

Implementation:

A specialized function named `CalculateAveragePriceFrom2016To2020` was added to the `Bond` class to determine the average price of the bond for the years from 2016 to 2020. This function utilizes the bond pricing logic and computes the average price based on prevailing interest rates for each year within the specified range.

The function first reads interest rate data from the `Bond_Ex3.csv` file. For each year in the range 2016 to 2020, it calculates the bond's price using the available interest rate and the bond's properties. Once all prices are computed, the function returns the average of these prices.

Test Results:

The `CalculateAveragePriceFrom2016To2020` function was thoroughly tested to ensure its correctness. The function was called from the main application, and the computed average price was printed to the command line for verification.

![Screenshot 2023-09-20 164950]

How to run the code:

Software - Requirements:

To run the code correctly you will require the following software components:

1. A Linux Terminal (it can be a native linux OS or a compatible environment like WSL on Windows).
2. g++ compiler (version: g+±11 or compatible).
3. make (version: GNU Make 4.2.1 or compatible).

Commands to run the code:

With the provided `Makefile`, running the code is streamlined. Follow these steps:

1. Open up a terminal in the directory containing the code.
2. Type in `make` and press enter. This will compile the code using the instructions from the `Makefile`.
3. Run the compiled binary using `./main`.

Alternatively, if you don't have `make` installed or prefer to compile manually:

1. Use the command `g++ main.cpp Bond.cpp -o main` to compile the code.
2. Run the compiled binary using `./main`.

If you wish to re-compile the code using `make`:

1. Use the command `make clean` to remove previous compiled files.
2. Then use `make` to compile again.