Final Project

Investing

You just got paid a $5,200 Holiday bonus, but because this year's holidays are scaled back, you decide to reinvest your money.

Since the US Government stimulus pushes the value of the dollar down, and since the stock market is at an all-time-high you decided to invest in Bitcoin. But now you are facing a dilemma…

1. Should you buy Bitcoin using all the money right away?
2. Or should you dollar-cost average that amount and buy $100 worth of Bitcoin every week for a year?

As a computer science student, you decide to rely on data while making this decision and write a program that will help you to make an informed decision.

For your calculations, you will rely on the historic 2019-2020 calendar year data.
In ***dbfile,*** you will find the records of bitcoin price taken every Saturday for 52 weeks.

Your Task

Write a program that will help you to decide on the strategy that you should follow and project the outcomes of both strategies:

1. Investing $100 every week

Write a function that will extract bitcoin price from the provided file, invest $100 every week, and calculates the amount of money you will have by the end of that year.
Please implement the following function:

```
double weeklyInvesting(string dbfile)
```

The function should return the worth of your investment in today's price. In order words, if you sell your accumulated bitcoin today, how much money would be worth.
The helper function for the opening and reading the file is provided for you at readFile();

2. Investing $5,200 all at once

You feel confident that you would be able to identify a relative deep in the Bitcoin price every given month. So, for the second function, you will extract the lowest price of Bitcoin for every given month and populate the price map with this data.

```
1 (January): $7351.53
2 (February): $9328.84
...
12 (December): $7112.61
```

Decomposing and creating a priceMap function would be a great idea!
dbByMonth file contains the annual price records split into 12 months to help you create this map and evaluate the lowest price of Bitcoin for every given month.

Once that is done, you will randomly choose one of these 12 price points and invest all $5,200 into Bitcoin based on that price. Implement the solution in the following function:

```
double investingOnce(string dbByMonth)
```

Same as the weeklyInvesting(), this function should return the worth of your investment in today's price.

3. Winning strategy

Once you have both functions working, give a conclusive answer. Which strategy performed better? For how much? What are the user's total gains?

```
void conclusion(string dbfile, string dbByMonth)
```

Solutuons

Solution 1

```
void conclusion(string dbfile, string dbByMonth){
  double purchasedWeekly = weeklyInvesting(dbfile);
  double purchasedOnce = investingOnce(dbByMonth);
```

```
  if (purchasedWeekly > purchasedOnce){
     cout << "You are better off buying Bitcoin weekly" << endl;
     cout << "You gained: $" << purchasedWeekly - purchasedOnce << " extra" << endl;
     cout << "Total gains: $" << purchasedWeekly - once << endl;
  } else{
     cout << "You are better off buying Bitcoin all in once" << endl;
     cout << "You gained: $" << purchasedOnce - purchasedWeekly << " extra" << endl;
     cout << "Total gains: $" << purchasedOnce - once << endl;
  }
}

double weeklyInvesting(string dbfile){
        double totalDollars = 0.00;
        double bitcoin = 0.000000;

        Vector<string> lines = readFile(dbfile);
        for (int i = 0; i < lines.size(); i++){
                double price = stringToDouble(lines[i]);
                bitcoin = weekly / price;
                double totalBitcoin += bitcoin;
        }
  totalDollars = totalBitcoin * bitcoinLatestPrice;
  return totalDollars;
}

double investingOnce(string dbByMonth){
        Map<double, double> priceMap;
        buildPriceMap(dbByMonth, priceMap);
        double key = randomInteger(1, 12);
        double price = priceMap[key];
        double bitcoin = once / price;
        double totalDollars = bitcoin * bitcoinLatestPrice;
        return totalDollars;
}

void buildPriceMap(string dbByMonth, Map<double, double> &priceMap){
        double price = 0.00;
        double key = 0;
        Vector<string> lines = readFile(dbByMonth);
```

```
        for (int i = 0; i < lines.size(); i++){
                if (i % 5 == 0){
                        key = stringToDouble(lines[i]);
                } else {
                        price = stringToDouble(lines[i]);
                        if (!priceMap.containsKey(key)){
                                priceMap[key] = price;
                        } else {
                                if (price < priceMap[key]){
                                        priceMap[key] = price;
                                }
                        }
                }
        }
}
```

Solution 2

The second solution demonstrates the different buildMap and investingOnce implementations while more straight forward functions conclusion() and weeklyInvesting() remain the same.

```
double investingOnce(string dbByMonth){
        double totalDollars = 0.00;
        double bitcoin = 0.000000;
        Map<double, Vector<double>> priceMap;

        priceMap = buildMap(dbByMonth);
         int key = randomInteger(1, 12);
        priceMap[key].sort();
        double price = priceMap[key][0];
        bitcoin = once / price;
        totalDollars = bitcoin * bitcoinLatestPrice;
        return totalDollars;
}

Map<double, Vector<double>> buildMap(string dbByMonth){
        Map<double, Vector<double>> priceMap;
        Vector<double> prices;
        Vector<string> lines = readFile(dbByMonth);
        double key = 0;
```

```
        for (string line : lines){
                double l = stringToDouble(line);
                if (l >= 1 && l <= 12){
                        key = l;
                } else {
                        double price = l;
                        priceMap[key].add(price);
                        if (prices.size() >= 4){
                                priceMap[key] = prices;
                                prices.clear();
                        }
                }
        }
  return priceMap;
}
```

Problem Motivation

Concepts Coverage

The problem is designed to test the knowledge of different collection types. The problem requires an understanding of the following concept:

1. Creation of Vectors and Vector operations
2. Creation and manipulation of Maps
3. Iterations over nested data structures
4. Passing and accessing data structures by reference
5. The use of strlib.h and math.h libraries

The problem also encourages decomposition and, once it is implemented, the student can come up with a very tidy and straightforward solution.

Personal Significance

The personal significance of this project is coming from two different areas, and I hope I combined both of them in this exercise.

First of all, the scope of the problem (buy at-once or spread over time) is highly debated in the financial world. To this day, this is one of the most googled questions when it comes to investing. Moreover, the volatility of the bitcoin market made it ten times more interesting as it could potentially lead to unexpected conclusions. Since I have been quite puzzled about this question myself, I loved the opportunity to research this topic for my final project.

Second, I deeply enjoyed working with sorting and all different kind of data structures throughout the course. Map, however, is still my favorite data structure due to its speed and its elegance. So I wanted to come up with a problem that could be effectively solved through the use of maps.

Common Misconceptions

The possible misconceptions or bugs could come from the following parts of the program:

1. Proper definition and use of data types

It is critical that students will use double as the main data type to store the price variable. It is even more critical to store in double the amount of bitcoin accumulated since that number will have 4-7 decimals after zero. The proper variable definition will lead to a neat solution, but the lack of doing so will lead to bugs and serious miscalculations of the results.

2. Proper unpacking of the data and conversion from the string to double

It would be important to remember that files store all the data in the string. After the file is read, the returning vector will store the price records in the string as well. Converting string price data into the double price data would be a critical step.

3. Building the map

It is apparent that students will correctly capture month pointers and would correctly store it as keys to the map. Moreover, students should either be able to sort the pricing data inside the map or compare the pricing data and only store the lowest price on the map. The second would be a more elegant solution but would require more mastery in map manipulation.