Instructions

In this lab you will build a simple simulator of the stock exchange, using hash tables with chaining to store, access, and modify stock prices efficiently. Companies in our fictitious stock exchange are denoted by one to four capital letters, for example “X”, “OF”, ”NFL”, or ”WNBA”, and transactions consist of the company name followed by the percentage change in its stock value. Here’s a simulated sample run, illustrating the desired functionality of your program:

Enter the number of transactions: 10
Initial value of all stocks: 100
Display transactions (Y/N)? Y
10 transactions will be processed:
E -0.62
ZS 1.00
M 2.18
YLBX -2.65
ZL -2.61
ZKV 0.67
KT 2.36
K 0.05
E -0.84
OF 0.54

The stock with the highest value is KT, with a value of $102.36
The stock with the lowest value is YLBX, with a value of $97.35
The stock with the most transactions is E, with 2

Data stored in a hash table of size 11. There are 9 items in the table, the longest list has length 2, load factor is 0.82, and there are 5 empty slots in the table. The table size was doubled 0 times.

Do you want to run the simulation again (Y/N)? N

The following main method was used to randomly generate the transactions. Use something similar for your assignment:

```java
public static void main(String[] args) {
    Random r = new Random();
    String S;
    double v;
    for(int i=0;i<10000;i++) {
        S = "";
        for(int k=0; k<=r.nextInt(4);k++)
```
Your task consists of implementing all the methods required to provide the functionality shown in the example above. Also, consider the following:

- Each node in the table needs to store the stock name, number of transactions, and stock value.
- You should keep track of the number of elements stored in the table (don’t count them repeatedly).
- Stocks that have not been part of any transactions don’t need to be stored in the table (their value is the initial value of all stocks ($100 in our example above), and their transaction counter is zero).
- Start with a table of 11 entries; replace the table by a table of size $(2n + 1)$ every time its load factor reaches 1.
- Since the key used for hashing is a string, you need to convert it to an integer value in a way that would ultimately result in as few collisions as possible.

Obviously, you are not allowed to use the built-in implementations of hash tables, you have to implement your own. Run experiments to compare the running times of your implementation for various numbers of transactions (for example, 100, 1000, 10000, and 100000). As usual, write a report explaining your work, as described in the syllabus.