VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

Assignment 2

FOREIGN EXCHANGE MARKET

(Part 2)

Ho Chi Minh City, 05/2020

Assignment 2 specification

Version 1.1

1 Outcomes

After completing this assignment, students will be able to:

- Implement the binary tree data structure.
- Perform basic operations on binary tree data structure.

2 Objective

Students are required to build a program on C ++ that reads information about the foreign exchange market [1] (part 4), implement simple requests on this information and write the result in a text file.
To accomplish this task, students need to download assignment2.zip file, which contains the files:

- src/main.h, src/main.cpp: Defining main functions, such as: read a text file containing information about the foreign exchange market, invoke function and write the results to the result file. Students are allowed to read these files but are not allowed to change any content of these files.
- src/processData.h, src/processData.cpp: Implementing information processing on currency transaction. Students perform tasks through this file. However, students are NOT allowed to add any #include other than the existing #include in these files.
- test/input/1.txt, test/input/2.txt: These file contain information about sample currency transactions, program inputs. Each file contains multiple lines, one for each currency transaction information processing command. The processing requirements for each order are detailed in Section 3.
- test/output/1.txt, test/output/2.txt: These file contain the result of processing currency transaction information corresponding to input file.

After extracting, students have to translate the provided C++ code, then execute by typing the following command on the command prompt (Command Prompt / Terminal):
main test/input/1.txt test/output/0.txt

After executing the above command, students check again by comparing the contents of the file **test/output/0.txt** and **test/output/1.txt** using the following command:

- Windows: *FC test/output/0.txt test/output/1.txt*
- Linux/MacOS:*diff test/output/0.txt test/output/1.txt*

The result of the comparison must show that these two files are not different.
Students do assignment 2 by editing the processData.h and processData.cpp files to organize data storage and information processing as required, test the program by creating new input on the input directory and the corresponding output file on the output directory.

3 Data processing command

3.1 General instruction

Each data processing order is a line on the currency transaction information file. Each command begins with a keyword (the word in bold in the description) and is followed by parameters (between < and > marks in the description). There is only one space between the command and the parameters. There are no spaces before the command keyword and no spaces after the final parameter. Some final parameters, which are placed in [and] in the description, are optional (appear or not appear in the command). When a command does not provide number of parameters exactly or does not have the same type of parameter as described or contains more extra spaces, the command will not be processed and the result will return -1. Otherwise, the command will be processed and return an integer value ≥ 0 as described in Section 3.2
The meaning of the acronym and the data type of the parameters is described as follows:

- TIME: an integer number (representing the time according to ISO standards) show the opening time of the session.
- BP: (bid price) a float number representing bid price.
- AP: (ask price) a float number representing ask price.
- BC: (base currency) a code (string) representing the purchase currency code.
- QC: (quote currency) a code (string) representing the sale currency code.
- LOT: (lot) a float number representing a value, which is calculated by lot.
- LV: (leverage) financial leverage in the form of: 1:X then LV = X.
- MN: (money) money in USD.
- ID: an integer representing the identifier of the transaction.
- INST: a string representing the command name keyword.

There are two optinal time's parameters in the command <TIME_A> and <TIME_B> (performing [<TIME_A> [<TIME_B>]]), the program will execute the command with other parameters and

- <TIME_A> ≤ <TIME> ≤ <TIME_B> if <TIME_A> and <TIME_B> are included in the command.
- <TIME> = <TIME_A> if there is only <TIME_A> in the command.
- Every <TIME> if there is no <TIME_A> and <TIME_B>.

Each order needs to be handled with a time complexity that does not exceed the rule in the "Complexity" column when N ≥ 100, where N is the average number of candles per currency exchange pair.

After finishing to process all instructions in the input file and write the results to the output file, your program must ensure to deallocate all dynamically allocated data objects and not to leave any memory garbage before ending.

3.2 List of commands

Requirement	Complexity	Description
SD <MN>	O(1)	Set up or change the deposit amount. In case of changing in the deposit amount, you must to comply with the restrictions in section 3.3. If the command executes successfully, returns 1, otherwise returns 0.
CD	O(1)	Check your deposit account and return the remaining funds in your account.
SL <LV>	O(1)	Set up or change your financial leverage. In case of changing in financial leverage, you must to comply with the restrictions in section 3.3. If the order executes successfully, it will returns the amount that the current user can exchange, otherwise returns 0.
INS <BC> <QC> <TIME> <BP> <AP>	O(log(N))	Add the buy/sell exchange rate of one currency pair at a time to the proposed tree data structure in Section 4. If the exchange rate already exists at the time we add the rate, we will update it with new data. If the command executes successfully, it will returns the TIME value of the root node in the tree data structure that is storing the pair's exchange rate of <BC>/<QC>.

DEL <BC> <QC> [<TIME_A> [<TIME_B>]]	O(log(N))	Delete data of exchange rate with <BC> and <QC> values correspond to the <BC>, <QC> parameters of the command according to the time provided by <TIME_A> and <TIME_B>. The command returns the TIME value of the root node in the tree data structure that is storing the pair's exchange rate of <BC>/<QC>.
UPD <BC> <QC> <TIME> <BP> <AP>	O(log(N))	Change the <BP>, <AP> values of the exchange rate with <BC>, <QC>, <TIME> values corresponding to the parameter. If there is no exchange rate data matching parameters <BC>, <QC>, <TIME>, the program will returns 0, otherwise, the program will changes the values <BP>, <AP> of the corresponding rate in the data structure and returns the TIME value of the root node in the tree data structure that is storing the pair's exchange rate of <BC>/<QC>.
OB <BC> <QC> [<TIME>] <LOT> or **OS** <BC> <QC> [<TIME>] <LOT> <ID>	O(log(N))	Open buy order (OB) or open sell order (OS) with the currency pair <BC>/<QC> at time <TIME> with the number of trading lot is ¡LOT¿. ¡ID¿ is a code to identify each transaction. The trade open order is successful when the ID does not match any of the previous trades and returns the corresponding traded amount. Otherwise, returns 0.
CB <BC> <QC> [<TIME>] <LOT> or **CS** <BC> <QC> [<TIME>] <LOT> <ID>	O(log(N))	Close buy order (CB) or close sell order (CS) with the currency pair <BC>/<QC> at time <TIME> with the number of trading lot is <LOT>. <ID> is a code to identify each transaction. The trade close order is successful when the ID of open order exists and returns the corresponding trade profit in USD. Otherwise, returns 0.

3.3 Constraints

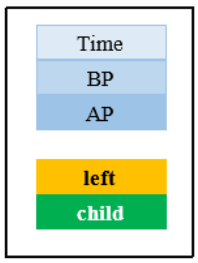
There are several constraints that need to be satisfied when processing these above commands:

1. It is not possible to set or change a financial leverage without setting up a deposit account.
2. You can only increase the amount in a deposit account if there are existing trades (buy/sell orders) that have not been closed.
3. Whenever, after the order is closed, if the deposit account is less than or equal to 0, all subsequent (buy/sell) transactions will not be made. All orders are not closed at the time of closing must be closed.
4. All profits not in USD must be converted into USD.
5. In this assignment, we will trade only USD currency pairs that exist in BC or QC.

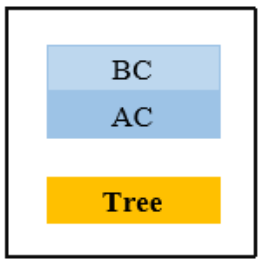
All commands that go against these constraints will return -1.

4 Recommended organization data structure

To organize the data for assignment, students require to use an AVL tree to implement the structure as below (Figure 2). As can be seen, because at each close and open trade order, we need to look for the exchange rate at that time to record. So AVL will be one of the most effective methods to help us search with complexity is always O(log (N)).



(a) Information of an exchange rate at a time



(b) Information of a currency pair

Figure 1: The archived information

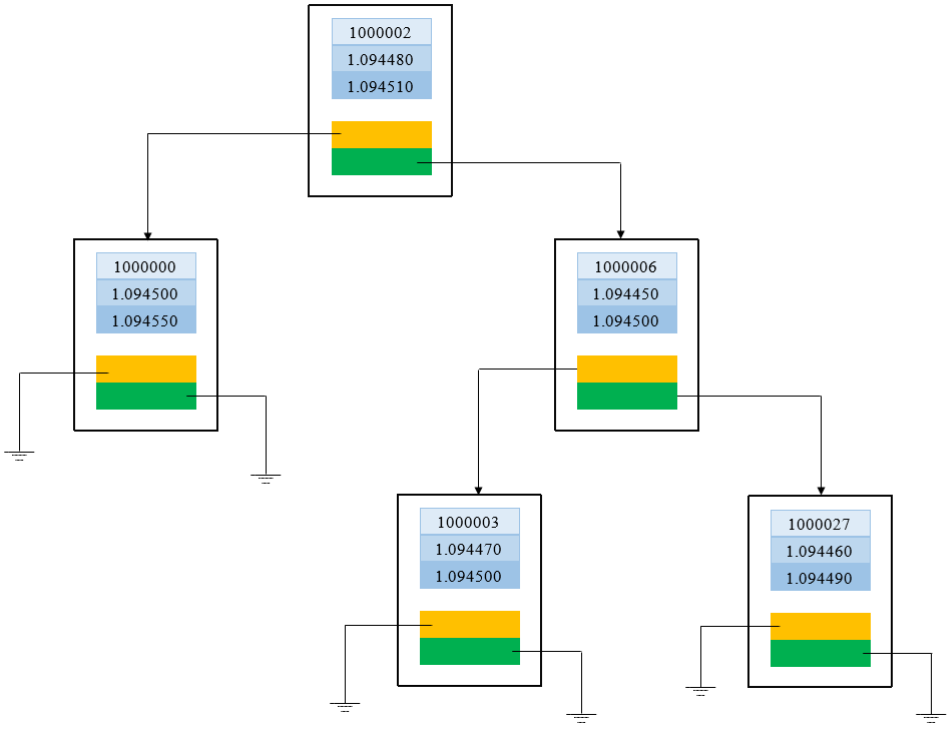


Figure 2: The exchange rate data of a currency pair

To implement the assignment, a single linked list should be used to store currency pairs information (Figure 1b), which contains the following information:

- BC
- QC
- Pointer points to an AVL tree (with a key is a value of TIME) to store the exchange information.

5 Submission

Students submit 2 files: **processData.h** and **processData.cpp** at "Assignment 2 Submission" in site "Cau truc du lieu và giai thuat (CO2003)_CC+CQ_(HK192)" on BKeL. The files must be in the plain form (NOT COMPRESSED) and have the correct name and can be translated successfully when combined with the provided main.h and main.cpp files. Again, in the processData.h and processData.cpp files, students cannot add any #include other than the #include available in these files.

Deadline for submission is announced at the place to submit. Just after the dealine, the link to submit will be closed automatically so you cannot submit anything then. To avoid any

risk, please upload your submission at least **one hour** before the dealine.

6 Plagiarism checking

The assignment must be DONE by yourself. Students will be considered cheating if:

- There is an unusual similarity between the source code of the submissions. In this case, ALL submissions are considered fraud. Therefore you must protect the source code of yourself
- Students do not understand the source code written by themselves, except for the initialized code. Students can refer to any resource, but make sure that you understand the meaning of all the commands you write. In case of not understanding the source code of the place where you refer, students are specifically warned that they should NOT use this source code; Instead, use what you have learned to write the program.
- Misleading submissions of other students on their personal accounts.

In the case of a conclusion of cheating, student score should be 0 for the entire course (not just only this assignment).

DO NOT ACCEPT ANY EXPLANATION AND NO EXCEPTIONS!

After each assignment is submitted, there will be a number of students who are randomly interviewed to prove that the assignment has been submitted by themselves.

References

[1] Tran Ngoc Bao Duy. *Assignment supporting document for Data structure and Algorithms - Foregin Exchange Market*, 2020.

——————————END——————————