Stock Market Portfolio Tracker

Aim

To develop a Stock Market Portfolio Tracker in C++ using various data structures such as linked lists, queues, stacks, trees, graphs, and hashing. The system manages stock prices, processes buy/sell orders, supports transaction rollback, represents company hierarchy, maintains investor connections, and enables fast stock ticker lookup.

Algorithm

1. Start the program.
2. Create a structure to store stock details and maintain them using a linked list.
3. Use a queue to store buy and sell order requests in FIFO order.
4. Process orders and store completed transactions in a stack.
5. Use the stack to rollback the most recent transaction if required.
6. Represent company hierarchy using a tree structure.
7. Represent investor relationships using a graph.
8. Use hashing to enable fast lookup of stock prices using ticker symbols.
9. Display stock prices, processed orders, and lookup results.
10. End the program.

Source Code

```
#include <iostream>
#include <list>
#include <queue>
#include <stack>
#include <unordered_map>
#include <vector>
using namespace std;
```

```
struct Stock {
   string ticker;
   double price;
};

list<Stock> stockList;

struct Order {
   string ticker;
   int quantity;
   string type; // BUY or SELL
};

queue<Order> orderQueue;

stack<Order> transactionStack;

struct Company {
   string name;
   vector<Company*> subsidiaries;
};

unordered_map<string, vector<string>> investorGraph;

unordered_map<string, double> tickerHash;

void addStock(string ticker, double price) {
   stockList.push_back({ticker, price});
   tickerHash[ticker] = price;
}

void placeOrder(string ticker, int qty, string type) {
   orderQueue.push({ticker, qty, type});
}

void processOrder() {
   if (orderQueue.empty()) {
      cout << "No orders to process\n";
      return;
   }
```

```
    Order o = orderQueue.front();
    orderQueue.pop();
    transactionStack.push(o);
    cout << o.type << " order processed for " << o.ticker
         << " Quantity: " << o.quantity << endl;
}

void rollbackTransaction() {
    if (transactionStack.empty()) {
        cout << "No transaction to rollback\n";
        return;
    }
    Order o = transactionStack.top();
    transactionStack.pop();
    cout << "Rolled back " << o.type << " order of " << o.ticker << endl;
}

void displayStocks() {
    cout << "\nStock Prices:\n";
    for (auto s : stockList)
        cout << s.ticker << " : " << s.price << endl;
}

int main() {

    addStock("TCS", 3850.50);
    addStock("INFY", 1620.75);

    placeOrder("TCS", 10, "BUY");
    placeOrder("INFY", 5, "SELL");

    cout << "Processing Orders:\n";
    processOrder();
    processOrder();

    rollbackTransaction();

    investorGraph["Arjun"].push_back("Priya");
    investorGraph["Priya"].push_back("Rahul");
```

```
    displayStocks();

    cout << "\nHash Lookup Price of TCS: "
         << tickerHash["TCS"] << endl;

    return 0;
}
```

Output

Processing Orders:

BUY order processed for TCS Quantity: 10

SELL order processed for INFY Quantity: 5

Rolled back SELL order of INFY

Stock Prices:

TCS : 3850.5

INFY : 1620.75

Hash Lookup Price of TCS: 3850.5
