A. Simplified Banking System

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Introduction:

Your task is to implement a banking system simulation using various design patterns. This problem is focused on creating a non-interactive system that processes a series of banking operations.

Requirements:

1. **System Features:**
   - **Account Creation:** Enable users to create accounts.
   - **Deposit and Withdrawal Operations:** Facilitate depositing and withdrawing funds.
   - **Transfer funds:** Users should be able to transfer money to other users
   - **Activate/Deactivate:** Users can deactivate and activate their accounts in instance of time.
   - **View Account Details:** Allow users to view their account details.
2. **Design Patterns:** Incorporate at least **three** of the covered design patterns that you think it makes sense. At least **one** design pattern from each category (Creational, Structural, Behavioral). The following design patterns are examples:
   - **Singleton:** Ensure only a single instance of the banking system exists.
   - **State:** Use to manage different states of an account (e.g., active, inactive, overdrawn).
   - **Strategy:** Implement for different operational strategies, like transaction fees for different account types.
   - **Adapter:** Allow interaction between incompatible interfaces, such as linking legacy systems with new account types.
   - **Decorator:** Use to add additional features to accounts dynamically (e.g., overdraft protection, rewards).
   - Note: You are not obliged to incorporate the suggested design patterns as long your choices fit their use. Make sure you choices are from what is covered

   Covered Design patterns:
   - Singleton
   - State
   - Prototype
   - Strategy
   - Adapter
   - Composite
   - Façade
   - Decorator
   - Proxy
   - Builder
3. **Account Types and (Transfer, Withdraw) Fees:** There are three types of accounts, each with its own transaction fee structure:

   a. **Savings Account:**
   - **Description:** Typically used for storing money not intended for daily expenses. Savings accounts often offer interest on the balance.

- **Transaction Fee:** 1.5% per transaction. This lower fee reflects the encouragement of saving and less frequent transactions compared to other account types.

b. **Checking Account:**

- **Description:** Designed for daily transactions and frequent access. These accounts usually don't earn interest and are used for regular expenses like bill payments.
- **Transaction Fee:** 2% per transaction. The slightly higher fee is due to the higher volume of transactions and the convenience of easy access and frequent use.

c. **Business Account:**

- **Description:** Tailored for businesses, handling higher volumes of transactions and offering features aligned with business needs, like payroll services or higher withdrawal limits.
- **Transaction Fee:** 2.5% per transaction. The highest fee among the accounts, justified by the higher transaction volumes and additional features provided for business purposes.

In each case, the transaction fee is deducted from the amount being withdrawn or transferred, directly affecting the transaction's net amount. For instance, transferring $100 from a Savings Account would result in a $1.5 fee, and the recipient would receive $98.5. These different fee structures and account characteristics cater to various user needs, from personal saving to business transactions.

4. **Allowed Operations:**

- Create
- Deposit
- Withdraw
- Transfer
- View(shows the initial deposit, deposit, withdrawals and transfers in order made from the account)
- Deactivate
- Activate

Note 1: that a person can't have two or more accounts at the same time.

Note 2: the initial state of the account is Active. During transferring only the account from which the transfer is made should have it in history. The account to which money is transferred, balance should be increased. Refer to the examples.

5. **Error Handling:** Systematically handle errors with specific messages with priority:

- **Account does not exist:** "Error: Account X does not exist."
- **Withdrawal from an inactive account:** "Error: Account X is inactive."
- **Insufficient funds for Transfer and Withdraw:** "Error: Insufficient funds for X."
- **Activate an activated account:** "Error: Account X is already activated."
- **Deactivate a deactivated account:** "Error: Account X is already deactivated."

Input

The input file is structured as follows:

- The first line contains a single integer n, where $1 \leq n \leq 2000$, denoting the number of banking operations.
- Each of the next n lines contains a banking operation command. The format for each command is `[T]N`, where T is the type of operation and N is the detail of the operation. Each parameter in the operation is denoted as `[type]parameterName`. Everything inside the parentheses "()" is an additional event description.

Accounts' names consist of lowercase and uppercase latin letters only. Numbers in the input, like $initialDeposit, will have either one digit after the decimal point, or will be without a decimal point. For example, the input might contain 100 or 250.5, but not 330.33.

1. "Create Account $[string]accountType $[string]accountName $[float]initialDeposit" ($\texttt{accountType}$ can be Savings, Checking, or Business. $1 \leq \texttt{initialDeposit} \leq 450$, $1 \leq \texttt{len(accountName)} \leq 10$).
2. "Deposit $[string]accountName $[float]depositAmount" ($1 \leq \texttt{depositAmount} \leq 450$).
3. "Withdraw $[string]accountName $[float]withdrawalAmount" ($1 \leq \texttt{withdrawalAmount} \leq 5000$).

	02:54	**45 баллов**
257235446	19.04.2024 02:43	Частичное решение: 3 баллов
257067427	17.04.2024 18:35	**Полное решение: 45 баллов**
256958336	16.04.2024 21:09	Частичное решение: 32 баллов
256957416	16.04.2024 21:00	Частичное решение: 32 баллов
256955680	16.04.2024 20:46	Частичное решение: 17 баллов
256665987	14.04.2024 18:17	Частичное решение: 17 баллов
256662807	14.04.2024 17:55	Частичное решение: 15 баллов

→ Набранные баллы

	Баллы
A	45
Всего	**45**

4. "Transfer $[string]fromAccountName $[string]toAccountName $[float]transferAmount" ($1 \leq \texttt{transferAmount} \leq 5000$).
5. "View $[string]accountName" (show details about a certain account).
6. "Deactivate $[string]accountName" (deactivate a certain account).
7. "Activate $[string]accountName" (activate a certain account).

Output

The ordered list of outputs matches the events in the input. Each event output is on its own line. Note that $transactionFeePercentage, the output should have only one digit after the decimal point, e.g. "1.5%". As for other float values like depositAmount, exactly three digits should be shown after the decimal point. Also note that each transaction and balance are in USD, and thus the symbol '$' should precede amounts in the output.

1. "A new $[string]accountType account created for $[string]accountName with an initial balance of $[float]initialDeposit."
2. "$[string]accountName successfully deposited $[float]depositAmount. New Balance: $[float]balance."
3. "$[string]accountName successfully withdrew $[float]withdrawalAmount. New Balance: $[float]balance. Transaction Fee: $[float]transactionFee ($[float]transactionFeePercentage%) in the system."
4. "$[string]fromAccountName successfully transferred $[float]transferAmount to $[string]toAccountName. New Balance: $[float]balance. Transaction Fee: $[float]transactionFee ($[float]transactionFeePercentage%) in the system."
5. "$[string]accountName's Account: Type: $[string]accountType, Balance: $[float], State: $[string]state, Transactions: $[TransactionEventList]history." ($state can either be Active or Inactive. The history of the transactions is saved without the fees. The history is shown by the order in which the transaction were carried out. An element in the history list can be one of the following: a) "Initial Deposit $[float]initialDeposit". b) "Deposit $[float]depositAmount". c) "Withdrawal $[float]withdrawalAmount". d) "Transfer $[float]transferAmount")
6. "$[string]accountName's account is now deactivated."
7. "$[string]accountName's account is now activated."

Examples

input Скопировать

```
5
Create Account Savings JohnDoe 450.0
Withdraw JohnDoe 150.0
Create Account Checking JaneDoe 300.0
Transfer JohnDoe JaneDoe 500.0
View JohnDoe
```

output Скопировать

```
A new Savings account created for JohnDoe with an initial balance of $450.000.
JohnDoe successfully withdrew $147.750. New Balance: $300.000. Transaction Fee:
$2.250 (1.5%) in the system.
A new Checking account created for JaneDoe with an initial balance of $300.000.
Error: Insufficient funds for JohnDoe.
JohnDoe's Account: Type: Savings, Balance: $300.000, State: Active, Transactions:
[Initial Deposit $450.000, Withdrawal $150.000].
```

input Скопировать

```
7
Create Account Savings JohnDoe 70.0
Create Account Checking JaneDoe 80.0
Deposit JohnDoe 30.0
Transfer JohnDoe Alice 1200.0
Create Account Savings Alice 50.0
Deactivate JohnDoe
Transfer JohnDoe Alice 1200.0
```

output Скопировать

```
A new Savings account created for JohnDoe with an initial balance of $70.000.
A new Checking account created for JaneDoe with an initial balance of $80.000.
JohnDoe successfully deposited $30.000. New Balance: $100.000.
Error: Account Alice does not exist.
A new Savings account created for Alice with an initial balance of $50.000.
JohnDoe's account is now deactivated.
Error: Account JohnDoe is inactive.
```

input Скопировать

```
8
Create Account Savings JohnDoe 70.0
```

```
Create Account Checking JaneDoe 80.0
Deactivate JohnDoe
Transfer JohnDoe Alice 20.0
Activate JohnDoe
Transfer JohnDoe Alice 1200.0
Create Account Savings Alice 50.0
Transfer JohnDoe Alice 1200.0
```

output Скопировать

```
A new Savings account created for JohnDoe with an initial balance of $70.000.
A new Checking account created for JaneDoe with an initial balance of $80.000.
JohnDoe's account is now deactivated.
Error: Account Alice does not exist.
JohnDoe's account is now activated.
Error: Account Alice does not exist.
A new Savings account created for Alice with an initial balance of $50.000.
Error: Insufficient funds for JohnDoe.
```

Note

Some useful links:

1. https://en.cppreference.com/w/cpp/io/manip/setprecision
2. https://en.cppreference.com/w/cpp/io/manip/fixed
3. https://en.cppreference.com/w/cpp/language/derived_class
4. https://en.cppreference.com/w/cpp/language/friend
5. https://en.cppreference.com/w/cpp/io/manip/setprecision