C++ Test - Transact (EFTS)

Greedy Algorithm:

The application running on an ATM can have multiple algorithms developed to choose the combination of bills from the bill cassettes to serve the customers. The simplest algorithm chooses the combination of highest denominations from the cassettes resulting in the minimum number of total bills. It is called Greedy algorithm. E.g.

If an ATM has 3 bill cassettes having 50 EUR, 20 EUR and 10 EUR bills.

- for 290 EUR, ATM dispenses 5 x 50 EUR and 2 x 20 EUR.
- for 160 EUR, ATM dispenses 3 x 50 EUR, 1 x 10 EUR.

During this exercise, you will be developing greedy algorithm for an ATM Software to dispense bills. Use console, GUI or file to take input of requested amount and display the combination of bills to be dispensed.

The structures which you must use are below;

```
typedef struct cdm_cu_info
{
        int Count;
        LPCASHUNIT *lppList;
} CUINFO, *LPCUINFO;
```

Description:

Count: The total number of cash units (structures) present in the ATM.

lppList: Pointer to an array of pointers to CASHUNIT structures below:

```
typedef struct cdm_cashunit
{
        int Type;
        char UnitID[5];
        short Denomination;
```

int Count;

int Maximum;

short Status;

*} CASHUNIT, *LPCASHUNIT;*

Description:

Type: Cash unit type i.e. {BILL=0, REJECT=1, RETRACT=2}. Only Bill cassettes dispense money.

UnitID: The unique alphanumeric identifier.

Denomination: Value of a bill in lowest denomination (cents). E.g. 5000 for 50 EUR bill. REJECT and RETRACT cassettes have 0 denomination value.

Count: Current count of bills in the cash unit.

Maximum: Maximum bills, a cash unit can offer in single dispense operation. It is defined as 0 for REJECT and RETRACT cassettes as well as if there is no maximum limit applied on bill cassettes.

Status: A cash unit can be {ONLINE=0, INOP=1}. Only ONLINE cash units take part in dispensing.

You should consider the following requirements while programming the solution.

1) Read the cash units values from an input file (use any format) to keep them persistent and configurable. Write a Parser function/class to fill the *cdm_cu_info* and *cdm_cashunit* structures with appropriate values. Update the structures in file after every dispense operation.
2) If ATM has no money or invalid configurations, put the ATM Out of Order with an appropriate message on screen.
3) After customer inputs an amount (from console, gui or input file etc.)
    a. If request is possible, show the combination of dispensable bills on screen.
    b. Present the bills (on screen) for a limited time e.g. 20 seconds and if customer does not take the money (simulate money taken event with a key/button press), retract the money back to the ATM. It means, the **Count** value of RETRACT (2) cassette is increased by the number of bills retracted.
        i. If RETRACT cassette is not ONLINE (0), then money shall not be retracted.
    c. If asked amount is not dispensable, show a proper error on screen and propose the next higher/lower amount possible. Examples below for the ATM with 3 bill cassettes having 50 EUR, 20 EUR and 10 EUR bills.
        i. If customer requests any amount below 10 EUR, propose on screen the higher possible 10 EUR if 10 EUR cassette has bills. Higher possible value changes if 10 EUR bills are not available.
        ii. If customer requests an amount between 20 EUR and 30 EUR, propose on screen both the lower possible 20 EUR if 20 EUR bills are available and higher possible

30 EUR if both 20 EUR and 10 EUR bills are available. Lower possible and higher possible values change if any cassettes are out of cash.

4) Show the current state of cash units on screen upon demand. The functionality is password protected. Password can be configurable in the file or hardcoded in the code.
5) Proper comments, exception handling and error logging also for the misconfigured input file.

Following questions should be answered in a separate text file

1) Write one sentence for each test case you tried to test your solution.
2) Explain in few words, what are the disadvantages of using greedy algorithm on an ATM to dispense money?
3) Suggest one or more other algorithms which you think should be better to develop in an ATM Software other than Greedy? Explain in few sentences how should they work?
4) What is the usage of REJECT cassette in the ATM during dispense operation? Explain in few words how could we simulate it in our designed solution?

Note:

- ATM can dispense only amounts possible to mix with denominations 5, 10, 20, 50, 100 EUR.
- The ATM can have minimum 1 cash unit and maximum 5 bill cassettes with 1 RETRACT and 1 REJECT cassette. An ATM will be Out of Order if these configurations are invalid.
- Consider that at any given point, one or more cash units can be in INOP state. Test your solution for relevant test case too.
- You are free to choose any OS and IDE to develop your solution. The solution is supposed to be developed and submitted in working condition. Submit the config file as well.
- The complete test should not take more than 3-4 hours to develop although you've 48 hours to submit the results back.