Lab #5

CS-2050

February 23, 2024

1 Requirements

As a note, unless *explicitly requested in the spec*, please **do not print anything** in the required functions. If you have debug printing which is not part of the lab specifications in your implementation file, please remove it before submitting. It makes grading your assignment more difficult when your submission unexpectedly prints during the grading process.

In this lab, you will be implementing a Vending Machine ADT. You are given a *partial typedef* in the starter code for this lab. You must complete the definition for the `VendingMachine` struct in your implementation function. Note that as the `VendingMachine` struct is to be defined in **lab5.c**, you will not be able to dereference it in **main.c**.

In this lab, you are given the following struct declarations:

```
// partial typedef, must complete in implementation file
typedef struct VendingMachine_t VendingMachine;

typedef struct {
    int ID;
    // The amount of this item currently in stock
    int stock;
    // The max amount of this item that can fit in a slot
    int maxStock;
    // The purchase price of 1 of this item
    float price;
} StockItem;
```

1.1 newMachine

```
VendingMachine * newMachine(int numSlots)
```

Info: This function creates a new, ***EMPTY***, vending machine with a number of item slots equal to *numslots*. Each slot must be able to hold an instance of `StockItem`. Note that "empty" slots have an ID of 0.

This function will return a pointer to the newly created vending machine on success, or NULL on failure. Your grade for this function will include your implementation of the `VendingMachine` struct type.

1.2 addStockItem

```
int addStockItem(VendingMachine *vm, StockItem item)
```

Info: This function takes a `VendingMachine`, and a `StockItem`. It inserts the item into the first empty slot in the vending machine. It returns 1 if insertion was successful, or 0 if the vending machine is has no empty slots left (or if insertion failed for any other reason). You may assume that no duplicate stock items will be inserted.

1.3 countExpensive

```
int countExpensive(VendingMachine *vm)
```

Info: This function takes a `VendingMachine`, and returns the number of items with a price $>=3$.

1.4 removeStockItem

```
int removeStockItem(VendingMachine *vm, int ID, StockItem *result)
```

Info: This function takes a `VendingMachine`, and the ID of a `StockItem`. If a stock item with the given ID exists in the vending machine, it will update the result pointer with the stock item to be removed, set the item slot to be empty, and return 1. If the ID is not associated with a stock item in the vending machine, it will return 0;

1.5 freeVendingMachine

```
void freeVendingMachine(VendingMachine *vm)
```

Info: This function takes a `VendingMachine`, and frees all memory allocated to it.

Grading: 23 points

1. Write required *newMachine* function
   * 10 points
2. Write required *addStockItem* function
   * 5 points
3. Write required *countExpensive* function
   * 2 points
4. Write required *removeStockItem* function
   * 5 points
5. Write required *freeVendingMachine* function
   * 1 points

Notice:

1. All of your lab submissions **must** include documentation to receive full points.
2. All of your lab submissions must compile under GCC using the $-Wall$, $-Werror$, and $-Wpedantic$ flags to be considered for a grade.
3. You are expected to provide proper documentation in every lab submission, in the form of code comments. For an example of proper lab documentation and a clear description of our expectations, see the lab intro PDF.