Lab #7

CS-2050

March 8, 2024

1 Requirements

Note that, although this week's lab looks very similar to lab 5, there are **important differences** you must take into account.

In this lab, you will be implementing a Vending Machine ADT using linked lists. 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 file. Note that as the `VendingMachine` struct is to be defined in **lab7.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()
```

Info: This function creates a new, empty, vending machine with a dynamic number of item slots. Each slot must be able to hold an instance of `StockItem`. **You are required to use a linked list as the primary backing data structure for your implementation in this lab.**

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 vending machine. It returns 1 if insertion was successful, or 0 if insertion failed. You may assume that no duplicate stock items will be inserted. You are not required to maintain any particular order of the inserted items.

1.3 countItems

```
int countItems(VendingMachine *vm)
```

Info: This function takes a `VendingMachine`, and returns the number of item slots currently in the machine. **Note that an item should be counted even if it is out of stock**.

1.4 countEmpty

```
int countEmpty(VendingMachine *vm)
```

Info: This function takes a `VendingMachine`, and returns the number of items with a stock equal to 0.

1.5 getStockItem

```
int getStockItem(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, and return 1. If the ID is not associated with a stock item in the vending machine, it will return 0;

1.6 freeVendingMachine

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

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

Grading: 27 points

1. Write required *newMachine* function
   * 10 points
2. Write required *addStockItem* function
   * 5 points
3. Write required *countItems* function
   * 2 points
4. Write required *countEmpty* function
   * 2 points
5. Write required *getStockItem* function
   * 5 points
6. Write required *freeVendingMachine* function
   * 3 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.