Purpose

Demonstrate the ability to implement and use structured data types utilizing the facilities of the C programming language.

Assignment

Write a program that simulates a soft drink machine. Your program will consist of two sections. One section will be the functions that make up the drink machine. The second section will be the functions that provide the user interface for the drink machine and allow the user to purchase drinks.

Section 1: The Drink Machine

This describes the operation of the first section of your program. This is the processing that is needed to simulate the Drink Machine.

There will be 1 or more drink types in the machine. Your program will read the number of drink types and the drink information from a file. You must then dynamically create an array of structures that will hold the drink information. You may assume that there are no errors within the input file.

The Drink Machine Input File Format

Here is a sample file that contains the information for one such Drink Machine:

```
8
Cola         1.25 25
Root-beer    1.25 20
Lemon-lime   1.25 25
Water        1.00 40
Orange       1.25  5
Iced-tea     1.25 35
Grape        1.30 15
Iced-Coffee  2.00 35
```

The first item is an integer that describes how many drinks are described in the input file.

Following that are the drink descriptions. The drink name is first, the cost of the drink is second and the number of drinks in the machine at start up is third. Note that the drink names do not include any spaces in the text.

You can use the C stream function fscanf() to read in the values from the file.

Code

You will be creating your Drink Machine code as a C style program. Your source will be in a file called `drinkmachine.c`. The structure definitions and function prototypes will be put into a file called `drinkmachine.h`.

The program will need a couple of structures.

***DrinkItem* structure**

One structure is for a ***DrinkItem*** and contains the following information:

id	The drink id (assigned by the program) of type int
name	Drink name (type of drink – read in from a file)
price	Drink cost (the retail cost of one drink). This is the price the customer will pay for one drink of this type. This is also read in from a file. Type is double.
number of drinks	Number of drinks of this type in the machine. Initial value is read in from a file. This is also updated by the program as people purchase drinks. Type int. Give this a good name.
Drinks purchased	Initially 0. Updated whenever a drink is purchased. Type is int. Give this a good name.

***DrinkMachine* structure**

The other structure, for the DrinkMachine, will contain the following information:

	An int that contains a version number. For this assignment this will have a value of 1.
	An int that contains the number of DrinkItem structures.
	An array of Drink Items. Each element of the array will be a DrinkItem structure. You will dynamically create this array based on the contents of an input file you will read in.
	An int that contains the current location in the array of DrinkItem structures. This is used internally by the drink machine part of your code.

You will choose the names for the structure items.

Functions

Your code will need to implement the following functions:

Function: *`create()`*

The first function you need to write for your Drink Machine is the `create` function.

This function takes no input parameters and returns back a pointer to your `DrinkMachine` structure.

First you need to dynamically create a `DrinkMachine` structure. You will return the address of this structure when you return from this `create` function.

You need to set the version number in the `DrinkMachine` structure to 1.

You also should create a global `const int` in your `drinkmachine.c` file. The name should be `INVALID_INDEX` and it should be given a value of -1. Initialize the current location value of the `DrinkMachine` structure to `INVALID_INDEX`.

Next you need to open up your input file. The input file is "`drink_machine.txt`". If the file does not exist you need to delete the `DrinkMachine` structure and return a `null` pointer back to the caller.

You need to read in the first number (8 in the example above) and create your `DrinkMachine` structure. You need to set the number of Drink Items to the number you just read in from the file (8 in the above example). NOTE: Your program must be able to handle any number of drinks. You cannot assume 8. You cannot assume any fixed number.

Next you need to create an array of `DrinkItem` structures. This array must be dynamically allocated using the number of Drink Items you read in above (8 in

our example). Again, you must be able to support any number of Drink Items. You must use dynamic memory allocation. **Do not rely on C99 style variable length arrays.**

You will need a loop that will execute once for every Drink Item. Inside the loop you will read in the information for a Drink Item, and put that information into your `DrinkItem` structure. You also need to set the drink id in the `DrinkItem` structure. It should have a value of 1 for the first drink, a value of 2 for the 2nd drink, etc.

Make sure your `create` function closes the input file once it has finished reading in the Drink Item information.

You function returns the address of the `DrinkMachine` structure to the calling function.

Function: *`destroy()`*

The `destroy` function has one input parameter, a pointer to a `DrinkMachine` structure.

Your `destroy` function should delete the array of `DrinkItem` structures you created. Finally, it should delete the `DrinkMachine` structure.

You must ensure that you are freeing up all memory you allocated in the `create` function.

Stop and Test:

Note: After you have written the `create` and `destroy` functions you could create a test program to call `create` and then call `destroy`. Your test code should be put into a file called `driver.c`. You can use the debugger to see that the structures and array of `DrinkItem` structures is being created properly. You can also use the debugger to make sure the destroy function is deleting the array of `DrinkItem` structures, and deleting the `DrinkMachine` structure. By doing this you can make sure the Drink Machine is being created and destroyed properly before you start writing addition functions.

Make sure you test the case where `create` returns a `null` pointer.

The firstDrink and nextDrink functions:

You will be creating two functions, `firstDrink` and `nextDrink`. They will be used to iterate through all of the drinks in the DrinkMachine.

You could just get access to the array and directly access the array elements from your driver program (the second part of your code). But what would happen if we decide to change how the `DrinkItem` structures are stored? We could change the program to use a vector. There are other ways of storing the `DrinkItem` structures that would not allow access via subscripts. We may want to use these in future versions of the DrinkMachine. To support these possible changes

in the future we are going to provide an interface in the Drink Machine that will work even if we change the way the `DrinkItem` structures are stored in some future version of the program.

The `firstDrink` and `nextDrink` functions will provide this interface.

Function: *`firstDrink()`*

The address of the `DrinkMachine` structure is passed to `firstDrink`. The `firstDrink` function returns back the address of a `DrinkItem` structure (or `null` pointer).

The `firstDrink` function needs to return back the address of the first of the `DrinkItem` structures in the array. If there are no array entries `firstDrink` should set the current location in the DrinkMachine structure to `INVALID_INDEX` and return back `null` pointer.

The current location in the `DrinkMachine` structure should be set to 0, the index of the first `DrinkItem` structure in the array. The current location is updated by the `nextDrink` function and is needed for it to work properly.

Function: *`nextDrink()`*

The address of `the DrinkMachine` structure is passed to `nextDrink`. The `nextDrink` function returns back the address of a `DrinkItem` structure (or `null` pointer).

The nextDrink function should get the next `DrinkItem` structure in the array, if there are any. The logic for `nextDrink` is as follows:

- If the current location in the `DrinkMachine` structure is `INVALID_INDEX` the function should return a `nullptr` to indicate that there are no more valid entries in the array. Note, this condition should only happen if `nextDrink` is called after all of the items in the array have been processed or if `nextDrink` is called and no `firstDrink` call has been made. This is a logic error in the programming using the `firstDrink` and `nextDrink` functions.

- Otherwise, the current location in the `DrinkMachine` structure should be incremented by 1. If it is less than the `DrinkMachine` structure's size value you still have a valid entry left and your program should return back the address of the `DrinkItem` structure at the new current location.

- If, after you incremented the current location in the previous step, the current location is now greater than or equal to the size in the `DrinkMachine` structure then you have a condition where there is no next `DrinkItem` structure in the array. You should set the current location in the `DrinkMachine` structure to

`INVALID_INDEX` and return a `null` pointer to indicate that there are no more valid entries in the array.

Stop and Test *`firstDrink()`* and *`nextDrink()`*:

Update your main function you used for earlier testing and test the new `firstDrink` and `nextDrink` functions. You can use the following logic (pseudo code):

Create `DrinkItem` structure pointer pDrink

For (pDrink = firstDrink(DrinkMachine structure); pDrink != nullptr; pDrink = nextDrink(DrinkMachine structure)
// Your processing goes here
End For

Make sure in your tests that all of the entries are being displayed properly. Also, try calling `nextDrink` without calling `firstDrink` and try calling `nextDrink` after the last entry has been retrieved and make sure your error handling works.

The remaining Drink Machine functions:

The remaining functions you need are for the following:

- Number of items. This returns back the total number of unique drink items. This will be the size of the array of DrinkItem structures in the DrinkMachine structure.
- Check to see if a drink item is available
- Determine the cost of a drink item
- Purchase a drink item
- A debugging function that displays the contents of the drink machine.

Function: *`items()`*

The `items` function has one parameter, the pointer to the `DrinkMachine` structure. It will return an `int`, the number of `DrinkItem` structures in the Drink Machine. For the current implementation this is the size of the array.

For the next 3 functions you need the drink id which is part of the `DrinkItem` structure.

Function: *`available()`*

The `available` function takes two input parameters. The first parameter is a pointer to the `DrinkMachine` structure. The second parameter is the drink id of a `DrinkItem` structure. The function checks to see if there if that drink is available (the quantity is 1 or more). The function returns back a `bool`. The value will be `true` if the drink is available and `false` if it is not available. The function will also return a `false` if the drink id is invalid.

Function: *`cost()`*

The `cost` function takes two input parameters. The first parameter is a pointer to the `DrinkMachine` structure. The second parameter is the drink id of a `DrinkItem` structure. The function will return back a value of type `double`. This will be the cost of that Drink Item. The return value will be negative if the drink id is invalid.

Function: *`purchase()`*

The `purchase` function takes four parameters and returns an `enum class`. The first parameter is a pointer to the DrinkMachine structure. The second parameter is the drink id of a `DrinkItem` structure. The third parameter is the amount of money the customer is using to purchase the drink. This is of type double. The fourth parameter needs to be passed by reference. This will be the amount of change, if any, to be returned to the customer. The fourth parameter is of type double.

The return value is an `enum class`. The name of the enum class should be `Purchase` with the values `PURCHASED`, INVALID, `NOT_AVAILABLE`, and `INSUFFICIENT_FUNDS`.

If the drink can be purchased the fourth parameter will contain the amount of change (0.0 if there isn't any change) and the function will return back `Purchase::PURCHASED`. The function also needs to decrement the quantity of the `DrinkItem` structure in the array. The function also needs to increment the purchased count for this item.

If the drink ID is not valid the function will return `Purchase::INVALID`.

If the drink id if valid but there aren't any drinks of that type left the function will return back `Purchase::NOT_AVAILABLE`.

Finally, if the drink id is valid and there are drinks of that type available, but the amount is insufficient to purchase the drink the function will return back a value of `Purchase::INSUFFICIENT_FUNDS`. The value of the fourth parameter will contain the cost of the drink (the same value returned from the `cost` function).

Function: *`dumpDrinkMachine()`*

The `dumpDrinkMachine` function takes a pointer to a `DrinkMachine` structure as the only parameter and does not return a value.

The function displays the contents of the drinks in the drink machine. The output should be similar to the following:

```
Id           Name Price  Qty Sold
 1           Cola  1.25   25    0
 2      Root-beer  1.25   20    0
 3     Lemon-lime  1.25   25    0
 4          Water  1.00   40    0
 5         Orange  1.25    5    0
```

```
6       Iced-tea  1.25   35    0
7          Grape  1.30   15    0
8    Iced-Coffee  2.00   35    0
```

Stop and Test the remaining functions:

Now that all of the functions needed for the Drink Machine have been written you need to test them. What should you test? You need to test the normal conditions and you need to test any error conditions.

Section 2: The Application

The second section of the code makes use of the drink machine functions you have already written. It should be created in a file named **drinkMachineDriver.c**.

You must infer its operation by looking at the output below. Here is sample output several runs of the final application. Your program should create the same output for the same input values. Your interface will look just like the samples runs shown here with the exception of the output from the `dumpDrinkMachine` function.

User input is in **bold**.

For the first run the user enters quit right away and no processing is performed.

```
Id            Name Price  Qty Sold
         1           Cola  1.25   25    0
         2      Root-beer  1.25   20    0
         3     Lemon-lime  1.25   25    0
         4          Water  1.00   40    0
         5         Orange  1.25    5    0
         6       Iced-tea  1.25   35    0
         7          Grape  1.30   15    0
         8    Iced-Coffee  2.00   35    0
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
0[Enter]
        Id           Name Price  Qty Sold
         1           Cola  1.25   25    0
         2      Root-beer  1.25   20    0
         3     Lemon-lime  1.25   25    0
```

```
         4          Water  1.00   40    0
         5         Orange  1.25    5    0
         6       Iced-tea  1.25   35    0
         7          Grape  1.30   15    0
         8    Iced-Coffee  2.00   35    0

Thank you for using the drink machine.
```

Here is another run with invalid amounts entered by the program's user. We also see change from one purchase and no change from another purchase:

```
        Id           Name Price  Qty Sold
         1           Cola  1.25   25    0
         2      Root-beer  1.25   20    0
         3     Lemon-lime  1.25   25    0
         4          Water  1.00   40    0
         5         Orange  1.25    5    0
         6       Iced-tea  1.25   35    0
         7          Grape  1.30   15    0
         8    Iced-Coffee  2.00   35    0
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
1[Enter]
Enter the amount for the purchase (up to $2.00): 1[Enter]
The amount you entered is insufficient to purchase the drink. The
price is 1.25
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
1[Enter]
Enter the amount for the purchase (up to $2.00): 2.01[Enter]
The amount entered is not valid.
Enter the amount for the purchase (up to $2.00): 1.3[Enter]
Your drink has been purchased. Your change is $ 0.05
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
```

```
          2      Root-beer  $  1.25
          3     Lemon-lime  $  1.25
          4          Water  $  1.00
          5         Orange  $  1.25
          6       Iced-tea  $  1.25
          7          Grape  $  1.30
          8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 1.25[Enter]
Your drink has been purchased.
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
          1           Cola  $  1.25
          2      Root-beer  $  1.25
          3     Lemon-lime  $  1.25
          4          Water  $  1.00
          5         Orange  $  1.25
          6       Iced-tea  $  1.25
          7          Grape  $  1.30
          8    Iced-Coffee  $  2.00
0[Enter]
         Id           Name Price  Qty Sold
          1           Cola  1.25   24    1
          2      Root-beer  1.25   20    0
          3     Lemon-lime  1.25   25    0
          4          Water  1.00   40    0
          5         Orange  1.25    4    1
          6       Iced-tea  1.25   35    0
          7          Grape  1.30   15    0
          8    Iced-Coffee  2.00   35    0

Thank you for using the drink machine.
```

And still another run. Here we show invalid values entered for the menu:

```
         Id           Name Price  Qty Sold
          1           Cola  1.25   25    0
          2      Root-beer  1.25   20    0
          3     Lemon-lime  1.25   25    0
          4          Water  1.00   40    0
          5         Orange  1.25    5    0
          6       Iced-tea  1.25   35    0
          7          Grape  1.30   15    0
          8    Iced-Coffee  2.00   35    0
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
          1           Cola  $  1.25
          2      Root-beer  $  1.25
          3     Lemon-lime  $  1.25
          4          Water  $  1.00
          5         Orange  $  1.25
          6       Iced-tea  $  1.25
```

```
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
```
9[Enter]
```
The drink id is not valid.
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
```
-1[Enter]
```
The drink id is not valid.
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
```
5[Enter]
```
Enter the amount for the purchase (up to $2.00): 
```
1.3[Enter]
```
Your drink has been purchased. Your change is $ 0.05
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
```
0[Enter]
```
        Id           Name Price  Qty Sold
         1           Cola  1.25   25    0
         2      Root-beer  1.25   20    0
         3     Lemon-lime  1.25   25    0
         4          Water  1.00   40    0
         5         Orange  1.25    4    1
         6       Iced-tea  1.25   35    0
         7          Grape  1.30   15    0
         8    Iced-Coffee  2.00   35    0

Thank you for using the drink machine.
```

One final (long) run. In this one we run out of Orange. The quantity for the Orange drink is 5. We successfully purchase 5 drinks, but we cannot purchase a sixth one:

```
    Id           Name Price  Qty Sold
     1           Cola  1.25   25    0
     2      Root-beer  1.25   20    0
     3     Lemon-lime  1.25   25    0
     4          Water  1.00   40    0
     5         Orange  1.25    5    0
     6       Iced-tea  1.25   35    0
     7          Grape  1.30   15    0
     8    Iced-Coffee  2.00   35    0
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 1.25[Enter]
Your drink has been purchased.
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 1.25[Enter]
Your drink has been purchased.
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 1.25[Enter]
Your drink has been purchased.
Enter a drink id for the drink you want to purchase or 0 to quit.
```

```
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 2[Enter]
Your drink has been purchased. Your change is $ 0.75
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 1.3[Enter]
Your drink has been purchased. Your change is $ 0.05
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
5[Enter]
Enter the amount for the purchase (up to $2.00): 2[Enter]
Sorry, we are out of your drink. Please select another drink
Enter a drink id for the drink you want to purchase or 0 to quit.
  Drink id          Drink    Price
         1           Cola  $  1.25
         2      Root-beer  $  1.25
         3     Lemon-lime  $  1.25
         4          Water  $  1.00
         5         Orange  $  1.25
         6       Iced-tea  $  1.25
         7          Grape  $  1.30
         8    Iced-Coffee  $  2.00
0[Enter]
        Id           Name Price  Qty Sold
         1           Cola  1.25   25    0
         2      Root-beer  1.25   20    0
```

```
3     Lemon-lime  1.25   25    0
4          Water  1.00   40    0
5         Orange  1.25    0    5
6       Iced-tea  1.25   35    0
7          Grape  1.30   15    0
8    Iced-Coffee  2.00   35    0

Thank you for using the drink machine.
```

Deliverables

- You must include your .c source files, your .h header files, and your input file all zipped up into a single .zip file.

Notes

- No late homework is accepted.
- To create a C program in Eclipse, go through the same process as you would for a new C++ project. Just make sure you select a new C Project. It will create a working HelloWorld for you.