

The Problem

The current soda vending machines lack the ability to keep track of income and inventory. Thus, the client would like a simulation of smart machine that will do the following:

- 1) Track revenue for machine
- 2) Maintain Inventory Levels of each soda
- 3) Block selection of any soda that is empty
- 4) Restrict payment amount to a max of \$1.00
- 5) Calculate change for customer
- 6) Report revenue

Specification

Purpose: To simulate a Soda Vending Machine

Input: A menu where user will select what type of soda they want.
Amount of money received from user.

Output: Change due
When user quits show total revenue.

Design

The following is a general pseudocode for this problem

```
Call function to load soda information (Name, price, Quantity on Hand)
Loop Selection Menu until Quit received
  Item Selected
    Request Payment
    Validate Payment (>0 .0 and <=1.00)
    Is Payment enough
      If no Request More Payment
    Make Change
    Add selection price to Total revenue
    Reduce selection Quantity by 1
On Quit entry
  Display total Revenue
```

The next section shows a more detail of each item in the General section

Load Soda Information

For each soda in machine

 Create new soda type (name, price, quantity)

Selection Menu

Display the following

1. *Soda 1*
2. *Soda 2*
3. *Soda 3*
4. *Soda 4*
5. *Soda 5*
9. *To Exit*

Input selection

If selection is 1 to 5

 Sale Soda #

If selection is 9

 Perform Exit Procedure

Sale Soda (Soda #)

Set payment to 0.00

While payment < Soda Price

 Display "Amount Paid: " payment

 Display "Please Enter Payment amount"

 Input newpayment

 If (newpayment + payment) > 1.00

 Display "Can not accept more than \$1.00"

 Else

 Payment = payment+newpayment

If payment > soda

 Display "Change = " payment – price

Subtract 1 from soda quantity

Revenue = revenue + price

Exit Procedure

Display "Revenue earned is " revenue

Implementation of the Design

```

struct Drink                                     // Data structure for each Drink
{
    string name;
    float price;
    int    quantityOnHand;
};
//Function Prototypes
void loadDrinkSelections(Drink *);
int mainDrinkMenu(Drink []);
double saleSoda(Drink&);
double receivePayment(double);
void makeChange(double, double);
void exitSimulation (double);
int main()
{
    // Local Variables
    Drink drinks[MAX_DRINK_CHOICES];
    double totalRevenue=0.0;
    int choice=0;
    bool isRunning = true;

    cout << fixed << showpoint << setprecision(2);

    //Load Soda Machine information
    loadDrinkSelections(drinks);

    //Start Machine
    while (isRunning)
    {
        choice = mainDrinkMenu(drinks);
        if (choice != 8)
            totalRevenue += saleSoda(drinks[choice]);
        else
            isRunning = false;
    }

    cout << "Total Revenue Earned $ " << totalRevenue << endl;
}
void loadDrinkSelections(Drink * array)
{
    array[0].name = "Cola";
    array[0].price = 0.75;
    array[0].quantityOnHand = 20;
}

```

```

        array[1].name = "Root Beer";
        array[1].price = 0.75;
        array[1].quantityOnHand = 20;

        array[2].name = "Lemon-Lime";
        array[2].price = 0.75;
        array[2].quantityOnHand = 20;

        array[3].name = "Grape Soda";
        array[3].price = 0.80;
        array[3].quantityOnHand = 20;

        array[4].name = "Cream Soda";
        array[4].price = 0.80;
        array[4].quantityOnHand = 20;
    }
    int mainDrinkMenu(Drink array[])
    {
        int ch;
        bool exitFlag = false;
        do
        {
            cout << "\nDrink Menu\n";
            cout << "-----\n";
            for (int x = 0; x < MAX_DRINK_CHOICES; x++)
            {
                int count = x + 1;
                cout << count << " - " << array[x].name << endl;
            }
            cout << "9 - to exit simulation.\n";
            cout << "-----\n";
            cout << "Enter choice: ";
            cin >> ch;
            if ((ch > 0 && ch < 6) || ch == 9)
                exitFlag = true;
            else
                cout << "Invalid Selection Try again!\n";
        } while (!exitFlag);

        return (ch-1);
    }
    double saleSoda(Drink &soda)
    {
        double output = 0.0, payment=0.0;

```

```

    if(soda.quantityOnHand > 0)
    {
        // sale soda
        payment = receivePayment(soda.price);
        makeChange(payment, soda.price);
        soda.quantityOnHand -=1;
        output = soda.price;
    }
    else
    {
        cout <<"Your selection is empty.\n PLease make a new Selection\n";
        output = 0.0;
    }

    return output;
}

double receivePayment(double cost)
{
    double payment=0.0;
    double newpayment=0.0;

    cout << "\nPrice $ " << cost << endl;
    do
    {
        cout << "Amount paid: " << payment << endl;
        cout << "Please enter payment amount: ";
        cin >> newpayment;
        if (newpayment+payment > 1.00)
            cout <<" Can not accept more than $1.00. \nYour payment of "
                << newpayment << " has been refunded.\n";
        else
            payment += newpayment;

    }while (payment < cost);

    return payment;
}

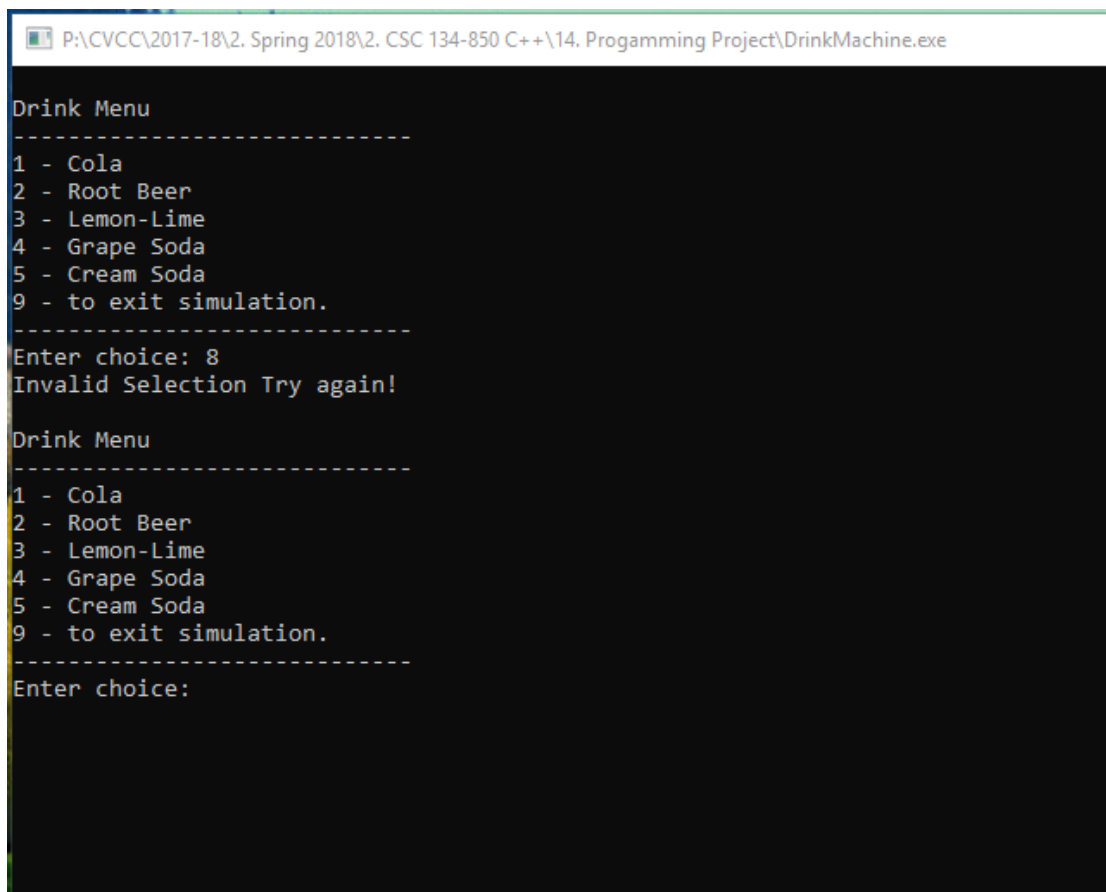
void makeChange(double paid, double cost)
{

```

```
        if (paid > cost)
            cout << "Your Change is " << (paid-cost) << endl;
    }
```

Testing of code

Tests were ran checking the main menu for proper selection input. If any number other than those listed an “Invalid selection” message was display.

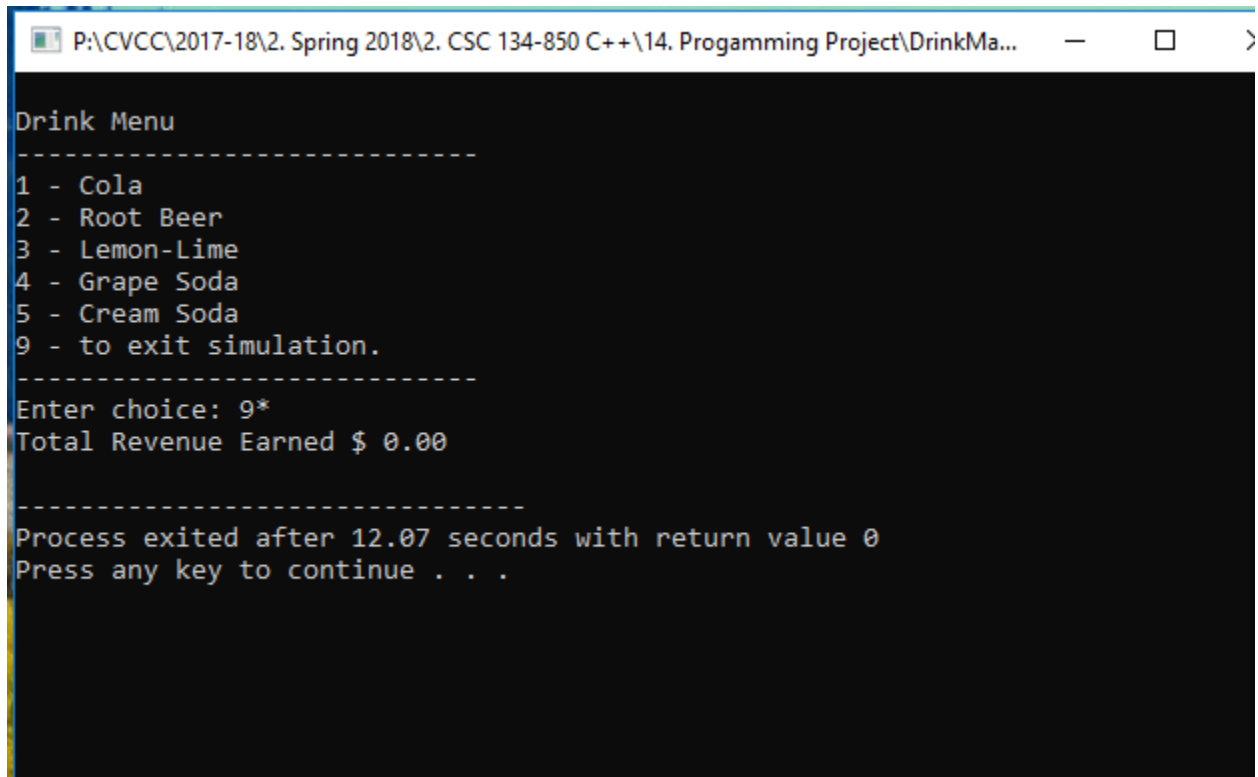


```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMachine.exe

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 8
Invalid Selection Try again!

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice:
```

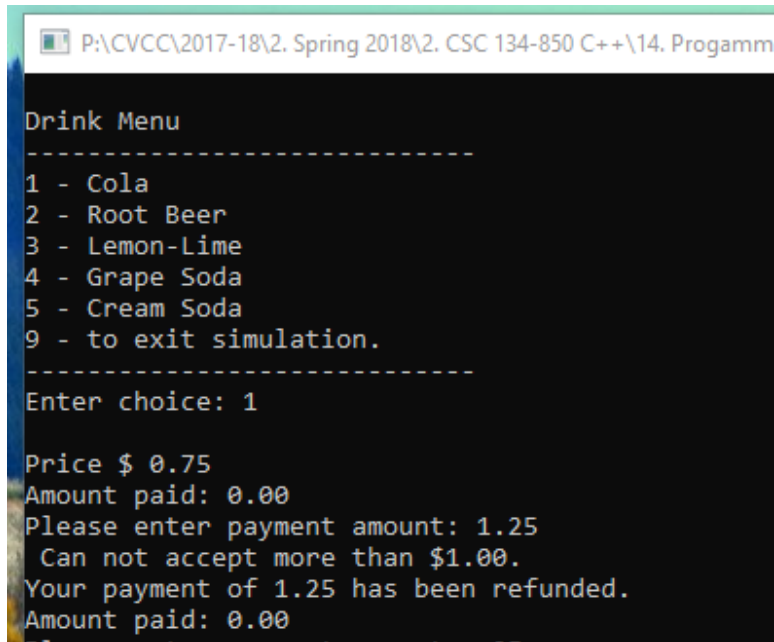
Also testing that the exit choice would provide required display before ending. The following display validates that this feature is also functional.



```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMa...
Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 9*
Total Revenue Earned $ 0.00

-----
Process exited after 12.07 seconds with return value 0
Press any key to continue . . .
```

With the menu function working properly, the next area to be tested was the Sale of soda area and related functions. This required running through each choice of soda, entering a payment amount, verifying the amount paid did not exceed the maximum allowed payment, making change if required, and then on exit show total revenue. The expected results for the purchase of one of each soda is \$3.85. The following screen captures will show the over payment and change functions are working properly.



```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programm

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 1

Price $ 0.75
Amount paid: 0.00
Please enter payment amount: 1.25
Can not accept more than $1.00.
Your payment of 1.25 has been refunded.
Amount paid: 0.00
```

Notice that the user entered an amount greater than \$1.00. Thus, the machine refunded the entire payment. When the payment is greater than the price but less than \$1.00 the program displays the change properly.


```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMac
Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 2

Price $ 0.75
Amount paid: 0.00
Please enter payment amount: .85
Your Change is 0.10

Drink Menu
-----
```

And finally, if payment amount is less than the price, the simulation requests additional payment.

```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMachine.
Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 3

Price $ 0.75
Amount paid: 0.00
Please enter payment amount: .5
Amount paid: 0.50
Please enter payment amount: .35
Your Change is 0.10

Drink Menu
-----
```

One problem surfaced that was not expected during testing. When the exact price was entered as payment, the system requested additional payment.

```
Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 4

Price $ 0.80
Amount paid: 0.00
Please enter payment amount: .80
Amount paid: 0.80
Please enter payment amount: .05
Your Change is 0.05

Drink Menu
-----
```

Upon further testing, this bug only affected a price ending with a zero, i.e. .60 or .80. The displayed numbers were expanded to eight places to the right of the decimal and the payment still equaled the cost. Thus, requiring additional payment. There is a work around that may solve this issue. Both price and payment amounts would have to be converted to an integer representing cents in lieu of dollars that is represented now. This conversion would have to happen twice for each payment cycle. Due to having to convert from dollars to cents and back, we currently do not recommend this option. A simpler option would be to add a small number, i.e. 0.00001, to the payment amount. This option will cause the "Your Change is" line to show on all transactions. Either of these resolutions will require client approval.

Finally, we checked all the total revenue function to see if total revenue was correct. This feature also performed properly.

```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMachine.exe

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 5

Price $ 0.80
Amount paid: 0.00
Please enter payment amount: .90
Your Change is 0.10

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 9
Total Revenue Earned $ 3.85
-----
```

With all these functions working properly, the next required item to test was when the quantity on hand for a selection was zero. To facilitate this feature, one of the selections was reduced to one unit. That selection was then chosen twice to verify that this requirement works properly.

```
P:\CVCC\2017-18\2. Spring 2018\2. CSC 134-850 C++\14. Programing Project\DrinkMachine.e

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 2

Price $ 0.75
Amount paid: 0.00
Please enter payment amount: .75

Drink Menu
-----
1 - Cola
2 - Root Beer
3 - Lemon-Lime
4 - Grape Soda
5 - Cream Soda
9 - to exit simulation.
-----
Enter choice: 2
Your selection is empty.
Please make a new Selection

Drink Menu
-----
```

Also, notice the exact payment feature worked properly here for the item priced at \$0.75. As mention earlier, the options to fixed the issue with the payment function will require the clients approval.