Understanding (for project)

- I need to create a program called carLot.cpp
  - It needs to define two structs
    - date
    - car
  - The main program loop should give the following options
    - add a car
    - print inventory
    - print profit for a given month/year
      - Dates must be validated, minus years and leap years
  - Do the above, and do it well.

Design

```
struct Date
{
}

Struct Car
{
}

void AddCar(takes a vector of cars)
{
        //asks all the questions needed to add a car
}

void CalculateProfit(takes a vector of cars)
{
        //asks the user for the date that they want

        //validates that date using the date validation function

        //subtracts sales from costs and prints
}

void PrintInventory(takes a vector of cars)
{
        //loops through the vector of cars and prints each element of the struct

        //if the car hasn't been sold, print N/A for the sale related elements

}

// Include prior code I have written to validate ints and doubles

bool ValidMonthDayCombo(int month, int day)
```

```
{
	//uses hard coded day-month associations to makes sure that the day actually could
	//exist in a given month
}

int main()
{
	//program loop using switch statement. Calls other functions.
}
```

Testing

Input	Expected output	output
1 Chevy Volt 2014 25000 2 28 2014 n 2 3 2 2014	Details of car, with N/A in sale related fields No cars sold in time period given	same

1 Chevy Volt 2014 25000 2 28 2014 n 1 Ford Ranger 2013 15000 2 30 // invalid date 2 5 2014 y 20000 2 10 2014 2 3 2 2014 4	Says invalid date Cars in inventory 1 car sold 5000.00 profit exits	same

Reflection

One thing that I struggled with is understanding why structs have to be limited to how the book describes them. Our exercises also limited them to plain-old-data. Technically a struct can contain methods (member functions). Why is this? My only guess is for backwards compatibility with C code?

Other than that, this week went fairly smoothly. I'm glad that we are getting more of the standard library and the STL available for our use. I was getting tired of implementing boilerplate code that other people have already implemented for me :). I'm excited to move on to classes and making my own data structures. This to me seems to be the meat and potatoes of OOP.