Homework #1
Due: Sunday, February 12th (11:59pm on Blackboard)

Programming: Follow proper coding standards as noted in class. *Submit a parent zipped folder with: a) a single PDF with all your programs and output copied and pasted in to the file, and b) all your C++ programs in their individual zipped folders.*

1. (40 points) Using the same "CarRecords.txt" input file from Blackboard, write an object-oriented C++ program in a single file called `CarRecords.cpp`. In the file, define two independent classes; `class Car`, `class CarRecords,` and the `main` function.

   Implement `class Car` with the following description:

	Car

`private:`	`string make`
	`string model`
	`int year`
	`string color`

`public:`	`Car()`
	`setFields(string mk, string md, int yr,string cl)`
	`string getMake()`
	`string getModel()`
	`int getYear()`
	`string getColor()`

Variables:

`make, model, year, color` – the four data fields from the file

Methods:

`Car()` – default constructor sets the data fields to their default values.

`setFields(``string mk, string md, int yr,string cl`**`)`** – sets the class member fields to the values passed.

`string getMake()` – returns make

`string getModel()` – returns model

`string getYear()` – returns year

`string getColor()` – returns color

Implement `class CarRecords` with the following description:

CarRecords	

`private:`	`int arraySize   // keep track of number of records`
	`ifsteam infile`
	`Car *cars`

`public:`	`CarRecords(int size) // Reads file records to array`
	`~CarRecords()`
	`void printCarRecords ()`
	`void sort_cars_by_make()`
	`void sort_cars_by_year()`
	`void print_duplicates()`

Variables:

`arraySize` – size of the array set to the number of records read from the file
`infile` – input file stream used to read information from a file
`cars` – a pointer object (to a dynamically allocated array of objects)

Methods:

`CarRecords(int size)` – constructor sets the value passed to the `arraySize` member, creates a dynamic array enough to hold `arraySize` objects of type `Car`. In addition, the constructor uses the `infile` file stream and the `setFields()` method to initialize all the `cars` array elements with the car records read from the file.
`~CarRecords()` – Destructor frees the memory allocated with `new`, and closes the file handler.
`void printCarRecords()` – prints out the car records from the array of objects (see sample output)
`void sort_cars_by_make()` – sorts the records in ascending order based on the **make** field.
`void sort_cars_by_year()` – sorts the records in descending order based on the **year** field.
`void print_duplicates()` – identifies any repeated records, and prints them out when found. Repeated records means that all the fields are the same.

Test your program with the main program below. If the user enters a value equal to or less than 10 for records to read, the program should create an arrays with that many elements dynamically and should only read that many records from the file, e.g. if the user enters 5, the program should create an array of 5 elements and only read 5 car records. If the user enters a number greater than 10, the program should create an array of only 10 elements and read all 10 records from the file. This checking is done in the `CarRecords(int size)` constructor when it reads the file.

Note: For modularity, or if needed, your `classes` and the `main` function can have other helper functions. Your code should be well commented.

```
int main() {
      int numRecs;
      cout << "Number or Records to read? " ;
      cin >> numRecs;
      CarRecords *cr = new CarRecords(numRecs);

      // Print car records
      cr->printCarRecords();
      // Sort by Year
      cr->sort_cars_by_year();
      // Print car records
      cr->printCarRecords();
      // Sort by Make
      cr->sort_cars_by_make();
      // Print car records
      cr->printCarRecords();
      // Check for Duplicates
      cr->print_duplicates();
      delete cr;
} // end main
```

Sample output

```
Number or Records to read? 15

PRINTING 10 RECORDS!
---------------------
Subaru, Outback, 2016, green
Toyota, Corolla, 2006, white
Dodge, Neon, 1993, pink
Ford, Fusion, 2013, yellow
Honda, Fit, 2015, blue
Ford, Expedition, 2009, silver
Toyota, Corolla, 2006, white
Ford, Fusion, 2013, yellow
Jeep, Cherokee, 1999, red
Mazda, Protoge, 1996, gold

SORTING RECORDS BY YEAR.....

PRINTING 10 RECORDS!
---------------------
Dodge, Neon, 1993, pink
Mazda, Protoge, 1996, gold
Jeep, Cherokee, 1999, red
Toyota, Corolla, 2006, white
Toyota, Corolla, 2006, white
Ford, Expedition, 2009, silver
Ford, Fusion, 2013, yellow
Ford, Fusion, 2013, yellow
Honda, Fit, 2015, blue
Subaru, Outback, 2016, green
```

```
SORTING RECORDS BY MAKE.....

PRINTING 10 RECORDS!
---------------------
Dodge, Neon, 1993, pink
Ford, Expedition, 2009, silver
Ford, Fusion, 2013, yellow
Ford, Fusion, 2013, yellow
Honda, Fit, 2015, blue
Jeep, Cherokee, 1999, red
Mazda, Protoge, 1996, gold
Subaru, Outback, 2016, green
Toyota, Corolla, 2006, white
Toyota, Corolla, 2006, white

CHECKING FOR DUPLICATES...
Ford, Fusion, 2013, yellow
Ford, Fusion, 2013, yellow
Toyota, Corolla, 2006, white
Toyota, Corolla, 2006, white
```

2. (60 points) Define a class hierarchy formed of a parent class `Furniture` and two child classes `Table` and `Bed`. Declare and define **each** of the three classes with a header (.h) file and a corresponding source (.cpp) file. The parent class should contain the following members:

- Three private floating-point numbers indicating the dimensions of the piece of furniture (`width`, `height`, and `depth`).
- A private string containing a unique name.
- A public constructor that takes the name of the piece of furniture as an argument.
- A public function `ReadDimentions`, used to read the values of the width, height, and depth from the keyboard. The function should show an error message if any of the entered values is less than 0.
- A public virtual function `Print()`, with no arguments and a `void` return value. This function should print the dimensions and name.
- The `Table` child class should have the following members:
- A private string indicating the type of wood used for the table, with two possible valid values: `"Pine"` and `"Oak"`.
- A public constructor that takes the unique name of the table as the first argument, passed directly to the constructor of the parent class, and the string corresponding to the wood type as the second argument. The constructor should print an error message if an invalid size string is passed.
- A public function `Print()` that overrides the function with the same name in the parent class. This function prints common furniture information by invoking the parent class function, and continues printing information specific to a table (the wood type).

The `Bed` child class should have the following members:

- A private string field specifying the size of the bed, with four possible valid values: `"Twin"`, `"Full"`, `"Queen"`, and `"King"`.
- A public constructor that takes the unique name of the bed as the first argument, passed directly to the constructor of the parent class, and the string corresponding to the bed size as the second argument. The constructor should print an error message if an invalid size string is passed.
- A public function `Print()` that overrides the function with the same name in the parent class. This function prints common furniture information by invoking the parent class function, and continues printing information specific to a bed (the size string).

In addition to the header (.h) files and a corresponding source (.cpp) files for each of the classes, provide an additional file `main.cpp`, which will include the main program. The main program should instantiate one object of type `Table` and another object of type `Bed`. Its properties should be populated based on values read from the user, and then printed with invocations to functions `Print()`. Provide a Makefile that compiles the programs. Create a zipped directory *assign2* for your programs.

This is an example of the execution of the program

```
Creating table...
  Enter name: MyTable
  Enter wood type (Pine, Oak): Pine
  Enter width: 1
  Enter height: 2
  Enter depth: 3
Creating bed...
  Enter name: MyBed
  Enter size (Twin, Full, Queen, King): Queen
  Enter width: 5
  Enter height: 6
  Enter depth: 7

Printing objects ...

MyTable:
  Width = 1, height = 2, depth = 3
  Pine wood
MyBed:
  Width = 5, height = 6, depth = 7
  Queen size
```