Homework #7 - Structs and Classes

Name:

1 Structs

One of the **big ideas** in C++ is something called OOP or **object oriented programming**. There are lots of real world programming problems that are best solved by putting a lot of data together into a new **type** that you make yourself. Grouping related data together can make your programs easier to design and easier to read and modify. In C++, you can group data together with a **struct**. A struct looks like this:

```
struct Student {
  string name;
  int age;
  float score;
};
```

structs.cpp

This struct would make it very easy to keep track of a list of students in an array, because now a single array can contain all this data together in one place (no need for a ”student names” array and a ”student ages” array, etc...). Creating an array to hold 3 students is now easy:

```
    // Make a struct that can hold 3 students
    Student students[3];
```

structs.cpp

Let’s say you wanted to change the name of the first student in the array, students[0]. You’d just write:

```
students[0].name = "Frank";
```

You could of course write a loop to fill in all the elements of the array, as well:

```
    for (int i = 0; i < 3; i++) {
        cout << "Enter name, age, and score..." << endl;
        cout << "Name: ";
        cin >> students[i].name;
        cout << "Age: ";
        cin >> students[i].age;
        cout << "Score: ";
        cin >> students[i].score;
        cout << endl;
    }
```

structs.cpp

2 Classes

Letting us group data into new types like this is a great feature, but C++ goes farther than that. It also allows us to put **functions** together with our data. Now, our data **and** the functions we need to work with it can all be in one place. We use classes to do this. Here's an example of the Student struct rewritten as a class:

```cpp
class Student {
    // Data
    private:
      string name;
      int age;
      float score;

    // Methods (functions inside the class)
    public:
      void setData(string selfName, int selfAge, float selfScore) {
          name = selfName;
          age = selfAge;
          score = selfScore;
      }

      void printData() {
          cout << "Name: " << name << endl;
          cout << "Age: " << age << endl;
          cout << "Score: " << score << endl;
      }
};
```

classes.cpp

Notice that the work of printing out everything inside the class can now be hidden inside the class. If I want to print out all the details for students[1] I would just say:

```cpp
students[1].printData();
```

To do this with a struct, I'd have to write:

```cpp
cout << "Name: " << students[1].name << endl;
cout << "Age: " << students[1].age << endl;
cout << "Score: " << students[1].score << endl;
```

Having functions saved inside with the data makes classes much more powerful than structs. There are lots of situations - in fact - where classes are very helpful because they can hide the details of their insides behind "set()" and "get()" functions, so programmers don't need to know what's inside a class in order to use it. These "get()" and "set()" functions can also make sure that anybody using the class is changing the class's data in the correct way.

3 Now you try...

3.1 Using Structs

Write a struct for use in a Zoo. You should make a struct that can store:

- An animal's name (ex: Billy)
- What it eats (ex: Bamboo)
- What type it is (ex: Panda)
- It's age (ex: 12)
- How many legs it has (ex: 4)

Once you have written this struct, you should make an array big enough to hold 3 animals, and make a loop to let the user enter the information for these 3 animals (name, age, legs, etc...). You can look at **structs.cpp** for an example.

3.2 Using Classes

Change your Zoo struct into a Zoo class. You should add functions to **set** the animal's name/-food/type/age/legs and another type to **get** these things and print them out. Start with the code in **classes.cpp**.

3.3 Final Challenge

Make a class called "Array" that can hold an array. The class should have functions to change what's stored in the array and print out the array. Something like this would be a good start:

```
class Array {
  // Data
  private:
    vector<int> array;

  // Methods (functions inside the class)
  public:
    void printArray() {
      // YOUR CODE HERE
    }

    void changeItem(int index, int newValue) {
      // YOUR CODE HERE
    }

};
```