CSCI 3110 (Barbosa)
Project 4: Fight the fire

Due: **See course calendar** – may not be turned late.

Assignment ID: **proj4**

File(s) to be submitted: **ffvehicle.h, ffvehicle.cpp, fftruck.h, fftruck.cpp, ffhelicopter.h, ffhelicopter.cpp, proj4.cpp, makefile**

Objectives: 1) Use inheritance to derive new classes; 2) Understand abstract base classes; 3) Use base class functions from a derived class; 4) Override base class virtual functions; 5) Define base class pure virtual functions; 6) Understand static and dynamic function call binding;

Project description:
In this assignment you will create a class representing a generic firefighting vehicle (*FFVehicle*), and two additional classes that inherit from this class, the *FFTruck* and *FFHelicopter* classes. The project is geared toward giving you practical experience with characteristics and C++ implementation details of inheritance and polymorphism.

Requirements:

1. Your program must be split into 7 files. There will be 3 classes (each with separate interface and implementation files), and a driver file. The requirements for these are specified below:

a) The *FFVehicle* class – This is an **abstract** class

- Files must be named ***ffvehicle.h*** and ***ffvehicle.cpp***
- Class must be named ***FFVehicle***
- Must contain #include guards for FFVEHICLE_H
- Must have these protected members
  i. A string containing the vehicle's name
  ii. A string containing the name of the fire's location
- Must have these public members
  i. A one parameter constructor that takes the vehicle's name.
  ii. A one parameter **virtual** void function, ***gotoFire***, with a string parameter representing the fire location. Sets the fire location data member and outputs:

  <vehicle name> ***responding to*** *<fire location>* ***fire.*** (see sample output)

  iii. A **virtual** void function, ***douseFire***, that outputs:

  <fire location> ***fire extinguished.*** (see sample output)

  iv. A parameterless **pure-virtual** void function, ***getWater*** that captures how each type of vehicle gets water to fight the fire.

b) The *FFTruck* class – This is a ***derived*** class that inherits from the *FFVehicle* class as public

- Files must be named ***fftruck.h*** and ***fftruck.cpp***
- Class must be named ***FFTruck***
- Must contain #include guards for FFTRUCK_H
- Has no private members

- Must have these public members
    i. A one parameter constructor that takes a string (the truck's vehicle name) – This constructor must pass this parameter to the base class constructor.
    ii. A function, ***getWater***, that **overrides/ implements** the pure virtual function in the base class and outputs:

    <vehicle name> ***hydrants connected.*** (see sample output)

    iii. A void function, ***douseFire***, that **overrides** the virtual function in the base class, and outputs:

    <vehicle name> ***pump pressure OK.*** (see sample output)

    Upon outputting this statement, this functions calls the *douseFire* function in the base class (which reports that the fire is extinguished)

c) The *FFHelicopter* class – This is a ***derived*** class that inherits from the *FFVehicle* class as public

- Files must be named ***ffhelicopter.h*** and ***ffhelicopter.cpp***
- Class must be named ***FFHelicopter***
- Must contain #include guards for FFHELICOPTER_H
- Has one private data member: a string that represents the helicopter's staging airport.
- Must have these public members
    i. A two-parameter constructor that takes a string for the helicopter's vehicle name, and a string representing the staging airport (in this order). This constructor passes the vehicle name parameter to the base class constructor and uses the second parameter to set its staging site data member.
    ii. A void function, ***gotoFire***, that **overrides** the virtual function in the base class, and: 1) sets the fire location name; and 2) outputs:

    <vehicle name> ***staging in*** *<stage site>* ***for*** *<fire location>* ***fire.*** (see sample output)

    iii. A function, ***getWater***, that **overrides/ implements** the pure virtual function in the base class and outputs:

    <vehicle name> ***scooping water from lake.*** (see sample output)

d) A driver, or client, file

- Must be named ***proj4.cpp***
- Must have two functions as shown below – *main* must be defined first
    - *fightFire* – This void function simulates the three stages of fighting a fire: 1) going to the fire (or staging site); 2) getting watter; and 3) dousing the fire with a single function call
        - It has two parameters: 1) a reference to a ***FFVehicle*** object; and 2) a string for the location of the fire.
        - It invokes three functions (with appropriate parameters) using the reference:
            1. *gotoFire*
            2. *getWater*
            3. *douseFire*
    - *main* – This function performs
        - Must be defined first
        - Opens and reads an input file named ***fires.txt***, in this format (see sample input file):
            1. Line 1: Name of the truck firefighting vehicle
            2. Line 2: Location name of fire to be fought by the truck
            3. Line 3: Name of the helicopter firefighting vehicle
            4. Line 4: Location name of fire to be fought by the helicopter
            5. Line 5: Name of the staging airport for the helicopter
        - Instantiates a *FFTruck* object with the information from the input file, and invokes the *fightFire* function (with the *FFTruck* object as an argument)
        - Instantiates a *FFHelicopter* object with the information from the input file, and invokes the *fightFire* function (with the *FFHelicopter* object as an argument)

2. Sample output (based on sample input file provided with this assignment) - Your output **must match**. Do not add additional verbiage, or change line spacing or character spacing (do not use tabs – only a single space between items). **Check your spelling**. See sample output below:

```
Truck1 responding to CityFire fire.
Truck1 hydrants connected.
Truck1 pump pressure OK.
CityFire fire extinguished.

Helo1 staging in LAX for ForrestFire fire.
Helo1 scooping water from lake.
ForrestFire fire extinguished.
```

3. Test your program - Use different input files.

4. Code comments - Add the following comments to your code:

- A section at the top of the source file(s) with the following identifying information:
  Your Name
  CSCI 3110-00X (your section #)
  Project #X
  Due: mm/dd/yy
- Below your name add comments in each file that gives an overview of the program or class.
- Place a one or two line comment above each function that summarizes the workings of the function.

5. Rubric

Requirement	Points Off
FFVehicle Class	
Class name	-5
Include guards	-3
Protected members	-5 (per member)
Constructor – number, types, and order of parameters	-10
Constructor – members correctly initialized	-5 (per member)
gotoFire – correct handling of fire location	-5
gotoFire – correct output	-5
douseFire – correct output	-5
getWater – correct pure virtual function signature	-10
FFTruck Class	
Class name	-5
Include guards	-3
Constructor – number, types, and order of parameters	-10
Constructor – correct call to base class constructor	-5
getWater – correct function signature	-5
getWater – correct output	-5
douseFire – correct function signature	-5
douseFire – correct output	-5
douseFire – correct call to base class *douseFire* function	-5
FFHelicopter Class	
Class name	-5
Include guards	-3
Private members declared/initialized per the specification	-5
Constructor – number, types, and order of parameters	-10
Constructor – correct call to base class constructor	-5

Constructor – derived class members correctly initialized	-5
gotoFire – correct function signature	-5
gotoFire – correct handling of fire location	-5
gotoFire – correct output	-5
getWater – correct function signature	-5
getWater – correct output	-5
Driver/Client	
fightFire – correct function signature/prototype	-10
fightFire – correctly invokes required functions	-5 (each function)
main – Defined first	-5
main – Name of input file	-5
main – Correctly reads input file	-5
main – Correctly instantiates *FFTruck* and *FFHelicopter* objects	-10 (each object)
main – Correctly invokes the *fightFire* function with each object as argument	-10
general – Does not compile (on *ranger* using Linux g++ compiler, and the submitted makefile)	-25
general – Runtime crash	-25
general – Other	TBD