Homework 8: Classes++

CS16 - Winter 2021

Due:	Thursday, March 4, 2021 (11:59 PM PST)
Points:	40
Name:	Tyler_Pruitt_____
Homework buddy:	_____

- You may collaborate on this homework with **at most** one person, an optional "homework buddy."
- **Submission instructions:** All questions are to be written (either by hand or typed) *in the provided spaces* and turned in as a single PDF on Gradescope. If you submit handwritten solutions write legibly. We reserve the right to give 0 points to answers we cannot read. When you submit your answer on Gradescope, **be sure to select which portions of your answer correspond to which problem** and clearly mark on the page itself which problem you are answering. We reserve the right to give 0 points to submissions that fail to do this.

1. (4 points) According to lecture and the textbook, what are the rules of class definition in order to make a class an abstract data type (ADT)?

```
According to the textbook and lecture, the rules of class definition in
order to make a class an abstract data type are

1) Ensure that all of the member variables of the class are private members

2) Ensure that every basic operation that the programmer or user will need
is a public member function, and to completely describe how to use or work
with each of of these public member functions

3) Ensure that any assisting or helping functions are private member
functions of the class
```

2. (2 points) What are derived classes and what mechanism do they use in order to fulfill what they need to do?

```
Essentially, derived classes are classes that stem from or are obtained from
another class. Derived classes inherit the member variables and functions
```

from the class that they are obtained from (their parent class). Derived classes use the mechanism of inheritance in order to fulfill what they need to do. Derived classes can be derived publicly or privately from their parent class. Derived classes that are derived publicly will have the same access level of the inherited member functions and variables as the parent class. However, derived classes that are derived privately will have all of the inherited member functions and variables be private.

3. (2 points) Can a derived class directly access by name a private member variable of the parent class?

No. A derived class cannot directly access by name a private member variable of the parent class. This cannot happen because even though the derived class is obtained from the parent class it still cannot directly access by name a private member variable of the parent class strictly because that member variable is private which means that any other class cannot directly access that member variable by name.

4. (2 points) Suppose the class `SportsCar` is a publicly derived class of a class `Automobile`. Suppose also that the class `Automobile` has public member functions named `accelerate` and `addGas`. Will an object of the class `SportsCar` have member functions named `accelerate` and `addGas`?

Yes. The class SportsCar will have the member functions named accelerate and addGas as public member functions as well because SportsCar was derived publicly from the class Automobile whose member functions accelerate and addGas also public. This works because derived classes that are derived publicly will have the same access level of the inherited member functions and variables as the parent class.

5. (14 points) Suppose your program contains the following class definition:

```
class Automobile {
   public:
      void set_price(double new_price);
      void set_profit(double new_profit);
      double get_price();
   private:
      double price;
      double profit;
      double get_profit();
};
```

Suppose the main part of your program contains the following declaration and that the program somehow sets the values of all the member variables to some values:

```
Automobile hyundai, jaguar;
```

Which of the following statements are then **not** allowed in the main part of your program and explain **why**.

(a) `hyundai.price = 4999.99;`
(b) `jaguar.set_price(30000.97);`
(c) `double a_price, a_profit;`
(d) `a_price = jaguar.get_price();`
(e) `a_profit = jaguar.get_profit();`
(f) `a_profit = hyundai.get_profit();`
(g) `if (hyundai == jaguar) {hyundai = jaguar;}`

(a) `hyundai.price = 4999.99;`
This is wrong because price is a private member variable of the class Automobile so it cannot be directly accessed by name outside of the class definition.

(b) `jaguar.set_price(30000.97);`
This is fine.

(c) `double a_price, a_profit;`
This is fine.

(d) `a_price = jaguar.get_price();`
It depends. If the variable a_price is declared before this line in the code then this statement is acceptable. Ideally, a_price should be declared as a double in this context. If the variable a_price is not declared before this line, then this statement is not valid and it cannot be placed in the main part of the program because the variable a_price has not been set up correctly.

(e) `a_profit = jaguar.get_profit();`
This is wrong because, assuming that a_profit has been correctly declared as a double, the member function get_profit() is a private member function of the class Automobile which means that it cannot be directly accessed outside of the class definition.

(f) `a_profit = hyundai.get_profit();`
This is wrong because, assuming that a_profit has been correctly declared as a double, the member function get_profit() is a private member function of the class Automobile which means that it cannot be directly accessed outside of the class definition.

(g) `if (hyundai == jaguar) {hyundai = jaguar;}`
This is wrong because the == operation is not defined for the class Automobile, thus the expression hyundai == jaguar is meaningless and will throw an error because the binary operation == makes no sense for these instances of this class.

6. (16 points) Suppose your program contains the following class definition:

```
class TwoNumbers {
   public:
      TwoNumbers(int n1, int n2);
      TwoNumbers(); // initializes num1, num2 to 0
      double sum(); // returns sum of num1 & num2
      double difference(); // returns diff. of num1 from num2
      double max(); // returns larger of num1, num2
   private:
      double num1, num2;
};
```

a. (10 points) Given the comments shown, give definitions to all 5 of these member functions/constructors:

```
TwoNumbers::TwoNumbers(int n1, int n2)
{
    num1 = n1;
    num2 = n2;
}

TwoNumbers::TwoNumbers()
{
    num1 = 0.0;
    num2 = 0.0;
}

double TwoNumbers::sum()
{
    return num1+num2;
}

double TwoNumbers::difference()
{
    return abs(num1-num2);
}

double TwoNumbers::max() {
    if (num1>=num2) {
        return num1;
    }
    else {
        return num2;
    }
}
```

b. (2 points) Consider these instructions in `main()`:

```
TwoNumbers thisOne, thatOne(5,7);
thisOne.num1++;
thisOne.num2 -= 7;
thatOne.num2 = thatOne.sum() + thisOne.difference();
cout << thisOne.max() / thatOne.max();
```

Explain all the reasons **why** this code will not compile.

The lines

```
TwoNumbers thisOne, thatOne(5,7);
cout << thisOne.max() / thatOne.max();
```

are perfectly acceptable. The first line correctly constructs two instances of the class TwoNumbers: thisOne and thatOne. The last line correctly prints out the result of the division of the max of thisOne by thatOne because the member function max() is a public member function of the class TwoNumbers so it can be called directly by name in main() as done here. This program will not compile because of the other lines of the code. This is because the member variables num1 and num2 are private member variables which means that num1 and num2 cannot be directly accessed by name in main() as done here. Therefore, the lines

```
thisOne.num1++;
thisOne.num2 -= 7;
thatOne.num2 = thatOne.sum() + thisOne.difference();
```

will not let the code complile because the statements .num1 and .num2 are not allowed in main() since num1 and num2 are private variables and as such they cannot be directly accessed, called, or changed in main().

c. (2 points) What would you change to the class definition to make this code compile?

In order to make this code compile, I would make the private member variables num1 and num2 public member variables so that they could be directly accessed or modified by name in main().

d. (2 points) When you fix it, what would these instructions do?

Once I have fixed the code, these instructions would first correctly construct the two instances of the class TwoNumbers: thisOne and thatOne. The num1 and num2 values of thisOne would initially be set to 0 and 0 respectively. The num1 and num2 values of thatOne would initially be set to 5 and 7 respectively. Then, the num1 value of thisOne will be incremented by one to the value of 1. Next, the num2 value of thisOne will be reassigned to the value of the expression (0 - 7) which is -7. Subsequently, the num2

value of thatOne will be reassigned to the value of the sum of num1 and num2 plus the difference between the num1 and num2 values of thisOne which is 20 (5+7 + |1-(-7)|). Finally, the maximum of the num1 and num2 values of thisOne (1, since num1,num2 for thisOne = (1,-7)) divided by the maximum of the num1 and num2 values of thatOne (20, since num1,num2 for thatOne = (5,20)) is printed out to screen which is (1/20) so 0.05 is printed out to screen since both the numerator and divisor are doubles (since both num1 and num2) are doubles.