Homework 7: Structs and Classes

CS16 - Winter 2021

Due:	Thursday, February 25, 2021 (11:59 PM PST)
Points:	50
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) Write a definition for a structure type for records consisting of a person's wage rate (dollars per hour), accrued vacation (in whole days), and status (hourly or salaried represented as either 'H' or 'S', respectively). Call the type `EmployeeRecord`.

```
struct EmployeeRecord{
    double wageRate;
    int accruedVacation;
    char status;
};
```

2. (6 points) Given the following structures defined:

```
struct Date {
   int day;
   int month;
   int year;
};

struct Person {
   string name;
   Date dateOfBirth;
};

struct ProjectTeam {
   Person MemberA, MemberB;
   Person Leader;
   string projectName;
   double projectBudget;
   Date projectDueDate;
};
```

If we declare `ProjectTeam TheATeam;` which was then initialized fully and correctly:

a. (2 points) How would you print (to standard out) the project budget for `TheATeam`?

```
cout << TheATeam.projectBudget << endl;
```

b. (2 points) How would you print (to standard out) the name of Member B of `TheATeam`?

```
cout << TheATeam.MemberB.name << endl;
```

c. (2 points) How would you print (to standard out) the year that the project leader of `TheATeam` was born?

```
cout << TheATeam.Leader.dateOfBirth.year << endl;
```

3. (5 points) What's the difference between a `struct` and `class` in C++?

In C++, a structure is an object without any member functions. A class, on the other hand, is a data type whose variables are objects themselves. Therefore, the difference between structures and classes in C++ is that structures are not allowed to have member functions. One could think of the difference between structures and classes as classes have member functions and structures do not.

4. (5 points) What's the difference between `public` and `private` members of a class in C++?

Public members of a class in C++ have no restrictions on them. However, private members of a class have the restriction that these members cannot be accessed directly in the program besides the exception of within the definition of the member function. This means that private member variables and private member functions are not able to be directly called or accessed outside of the definition. Thus, for the rest of the program these private members of a particular class are "hidden" from the program and only the public members of that class are directly accessible to the rest of the program, including definitions of any function.

5. (5 points) What are class constructors?

Class constructors are particular kinds of class specific member functions that is automatically called when an object of this class is declared and it is used to initialize the values of the member variables and finish up any other initialization of that particular object of that class. Essentially, class constructors are called automatically when an object of a particular class is declared and the class constructors initialize the object. The reason they are called class constructors is because that is what they fundamentally do, they construct classes.

6. (25 points) Suppose your program contains the following class definition:

```
class Point {
   public:
      Point(double n1, double n2);
      Point(); // initializes member variables to 0
      double get_x(); // returns value of x
      double get_y(); // returns value of y
      void set_x(double n); // sets a new value for x
      void set_y(double n); // sets a new value for y
   private:
      double x, y;
};
```

a. (12 points) Given the comments shown, give definitions to all 6 of these member functions/constructors:

```
Point::Point(double n1, double n2) {
    x = n1;
    y = n2;
}

Point::Point() {
    x = 0;
    y = 0;
}

double Point::get_x() {
    return x;
}

double Point::get_y() {
    return y;
}

void Point::set_x(double n) {
    x = n;
}

void Point::set_y(double n) {
    y = n;
}
```

For points $(x_1, y_1), (x_2, y_2)$, the Euclidean distance formula is given by:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Suppose we want to add a member function to the `Point` class that computes the distance between a given point and itself. Call it `distanceFrom`. The function should take as argument *another* object of type `Point` and return the computed distance. Assume that the `<cmath>` library is already included.

b. (2 points) Give the member function *declaration* for the `distanceFrom` member function.

```
double distanceFrom(Point otherPoint);
```

c. (4 points) Give the member function *definition* for `distanceFrom`.

```
double Point::distanceFrom(Point otherPoint) {
    double dist, distSquared;

    distSquared = pow((otherPoint.get_x() - x), 2) + pow((otherPoint.get_y() - y), 2);

    dist = pow(distSquared, 0.5);

    return dist;
}
```

For a point (x, y), we can rotate it by θ degrees to obtain a new point (x', y'):

$$x' = x\,cos(\theta) - y\,sin(\theta)$$
$$y' = x\,sin(\theta) + y\,cos(\theta)$$

Suppose we want to add a member function to the `Point` class that rotates the point by a given degree and *updates* the values for the member variables `x` and `y`. Call it `rotate`. The function should take as argument a double representing the degree θ. Assume that the `<cmath>` library is already included.

d. (2 points) Give the member function *declaration* for the `rotate` member function.

```
void rotate(double deg);
```

e. (5 points) Give the member function *definition* for `rotate`.

```
void Point::rotate(double deg) {
    x = x*cos(deg) - y*sin(deg);
    y = x*sin(deg) + y*cos(deg);
}
```