Homework8 – Spring 2023 (Due: May 31, 2023)

In this homework, you'll explore more concepts from OOP palooza. The required problems touch mainly on different forms of inheritance, polymorphism, and dynamic dispatch. Some optional problems touch on course topics that are important, but not directly discussed in lecture (and thus not testable): abstract classes, details on overloading, and SOLID. Some questions have multiple, distinct answers which would be acceptable, so there might not be a "right" answer: what's important is your ability to justify your answer with clear and concise reasoning that utilizes the appropriate terminology discussed in class. Each question has a time estimate; you'll know you're ready for the exam when you can solve them roughly within their time constraints.

We understand, however, that as you learn these questions may take more time. For that reason, only **starred red** questions need to be completed when you submit this homework (the rest should be used as exam prep materials). Note that for some multi-part questions, not all parts are marked as red so you may skip unstarred subparts in this homework.

You must turn in a PDF file with your answers via Gradescope - you may include both typed and handwritten solutions, so long as they are legible and make sense to our TAs. Make sure to clearly label each answer with the problem number you're solving so our TAs can more easily evaluate your work.

1. ** Consider the following class and interface hierarchy:

```
interface A {
 ... // details are irrelevant
}

interface B extends A {
 ... // details are irrelevant
}

interface C extends A {
 ... // details are irrelevant
}

class D implements B, C
 ... // details are irrelevant
}

class E implements C
 ... // details are irrelevant
}

class F extends D {
 ... // details are irrelevant
}

class G implements B {
 ... // details are irrelevant
}
```

a) ** (5 min.) For each interface and class, A through F, list all of the supertypes for that interface or class. (e.g., “Class E has supertypes of A and B”)

b) ** (3 min.) Given a function:

```
void foo(B b) { … }
```

which takes in a parameter of type B, which of the above classes (D - G) could have their objects passed to function `foo()`?

c) ** (1 min.) Given a function:

```
void bar(C c) { … }
```

Can the following function `bletch` call `bar`? Why or why not?

```
void bletch(A a) {
  bar(a);  // does this work?
}
```

3. ** (5 min.) Explain the differences between inheritance, subtype polymorphism, and dynamic dispatch.

4. ** (5 min.) Explain why we don't/can't use subtype polymorphism in a dynamically-typed language. Following from that, explain whether or not we can use dynamic dispatch in a dynamically-typed language. If we can, give an example of where it would be used. If we can't, explain why.

5. (5 min.) Consider the following classes:

```
class SuperCharger {
public:
  void get_power() { ... }
  double get_max_amps() const { ... }
  double check_price_per_kwh() const { ... }
};

class ElectricVehicle {
public:
  void charge(SuperCharger& sc) { ... }
};
```

Which SOLID principle(s) do these classes violate? What would you add or change to improve this design?

Note: we didn't discuss SOLID in lecture; peep Carey's slides for a quick overview!

6. (10 min.) Does the Liskov substitution principle apply to dynamically-typed languages like Python or JavaScript (which doesn't even have classes, just objects)? Why or why not?

   Note: we didn't discuss SOLID in lecture; peep Carey's slides for a quick overview!