PIC 10A 1C. Week 9b Exercises. TA: Eric Kim.

Note: In this class, always make constructors and member functions public, and member variables private! Also: const-correctness, and reference parameters when appropriate.

1. Oh my glob

```
class Person {
public:
  Person(string name, int age);
  Person(string name) : myname(name), myage(1000) {}
  Person(int age) : myname("Glob"), myage(age) {}
  string get_name() const;
  int get_age() {  return this->myage; }
private:
  string myname;
  int myage;
};
Person::Person(string name, int age) : myname(name), myage(age) {}
string Person::get_name() const {
  return this->myname;
}
```

Consider the following code. Find the errors, and describe what went wrong:

```
    Person finn("Finn", 12);
    const Person jake("Jake", 30);
    cout << finn.get_name() << " " << jake.get_name();
    cout << finn.get_age() << " " << jake.get_age();
    cout << jake.myage + finn.myage;
    Person marcy("Marceline");
    Person glob(9000);
```

2. Even Steven meets Steven Evens

Write a class EvenGenerator that generates the even numbers starting from 2.

```
  EvenGenerator mygen;
  cout << mygen.next(); // Displays: 2
  cout << mygen.next(); // Displays: 4
  int val = mygen.next();
  cout << mygen.next(); // Displays: 8
```

3. Expand your mind

Write a function expand_nums that, given a vector of positive ints, repeats each integer based on its value. Negative numbers and zero should be unmodified:

```
  vector<int> nums = {2, 1, 0, 3, -5};
  vector<int> out = expand_nums(nums); // out is: [2,2,1,0,3,3,3,-5]
```

4. Hip to be square

Consider the following function vecsquare:

```
/**
 * Squares the integers of a vector. Modifies (mutates) the vector.
 * @param vec The input vector of integers.
 * @return void
 **/
void vecsquare(vector<int> vec) {
  for (size_t i = 0; i < vec.size(); ++i) {
    vec[i] = vec[i]*vec[i];
  }
}
```

The desired usage is:

```
    vector<int> v = {2, 3, 4};
    vecsquare(v); // v is now: [4, 9, 16]
```

Does the function vecsquare behave correctly? If not, describe why not, and suggest a simple fix.

5. Saving Pvt. Private (starring: Maj. Major)

```
class A {
public:
  A() : b(0) {}
  void foo();
private:
  int b;
};
void A::foo() {
    this->b = this->b + 10;
}
```

The member variable "b" is declared private, yet foo() accesses and modifies b. Does foo() compile correctly? Explain why or why not.

6. Shadowing this

Recall that, within a class, "this" is a pointer to the current object. However, using it is sometimes optional when accessing member variables. For instance, the following is valid code:

```
class Dog {
public:
  Dog(string name) : myname(name) {}
  void bark() const;
private:
  string myname;
};
void Dog::bark() const {
  cout << this->myname << endl;
  cout << myname << endl; // This works too!
}
```

What is a scenario where you must use the "this" parameter to refer to the object's member variable?

7. Don't fib around

Write a class FibGenerator that generates the Fibonacci sequence: 1, 1, 2, 3, 5, 8, 13, 21, etc. Recall that the n-th term of the sequence is simply the sum of the previous two terms:

```
FibGenerator myfib;
for (int i = 0; i < 8; ++i) {
  cout << myfib.next() << " ";
}
// Outputs: 1, 1, 2, 3, 5, 8, 13, 21
```

Hint: You may want to keep track of the previous two terms by defining two member variables.