CECS 282 - Homework 12

Complete these problems on a separate sheet of paper. Due the day of your final.

1. Reading from *C++ How to Program*:
   (a) Chapter 12.5
   (b) Chapter 10.12 (review)
   (c) Chapter 17.9
   (d) Chapter 24.2 (`shared_ptr` only)

2. The following code fragment from Homework 8 has a subtle memory leak in it:

```
try {
   cout << "Enter a number less than 10" << endl;
   int *x = new int;
   cin >> *x;
   if (*x >= 10)
      throw std::out_of_range("listen to the instructions dummy");
   delete x;
}
catch (std::out_of_range &ex) {
   cout << "You didn't listen!"  << endl;
}
```

   Using the `std::unique_ptr` class discussed in lecture, rewrite the example above so that the `new int` is wrapped in a `unique_ptr`, and ensure that even if the `out_of_range` is thrown, the heap integer will be deleted. Hint: you should have **no** `delete` statements!

3. Using the `std::accumulate` function discussed in lecture, give the output of the following code fragment: For each call to `std::accumulate`, describe what the net calculation of the call is in terms of the vector used as a parameter. One example is given.

```
int Add(int a, int b) { return a + b; }
int Min(int a, int b) { return a <= b ?  a :  b; }
bool Any(bool a, bool b) { return a || b; }
bool All(bool a, bool b) { return a && b; }

int main() {
   vector<int> values = {8, 6, 7, 5, 3, 0, 9};
   vector<bool> flags = {true, true, true, false, true, false};

   cout << std::accumulate(values.begin(), values.end(), 0, Add);
   // EXAMPLE ANSWER: outputs 38, which is the SUM of all the numbers.


   cout << std::accumulate(values.begin(), values.end(), 2147483647, Min);
   cout << std::accumulate(flags.begin(), flags.end(), false, Any);
   cout << std::accumulate(flags.begin(), flags.begin() + 3, true, All);
}
```

4. Given the following function:
   `int Pow(int a, int b); // assume this function returns` a^b.

   Show how to declare a `std::function` variable named `func` that points to `Pow`, then give the output of the following line of code:

```
cout << func(func(3, 2), func(4, 1));
```

5. In the following code, what type will be inferred for each of the six `auto` variables declared?

   (a) `auto a = 10;`

   (b) `auto b = 10L;`

   (c) `auto c = "hello";`

   (d) `auto d = new Rational(1, 3);`

   (e) `auto e = 1 / 2;`

   (f)
   ```
   vector<int> values = {1, 2, 3};
   auto itr = values.begin();
   ```

6. **All** of the following C-style casts will compile in C++. Which of them would **not** compile if replaced by `static_cast`?

   (a)
   ```
   double g = 8.5;
   int i = (int)g;
   ```

   (b)
   ```
   double g = 8.5;
   int *i = (int*)(&g);
   ```

   (c)
   ```
   string s = "8.5";
   double *p = (double*)(&s);
   ```

   (d)
   ```
   class A {}; class B : public A {}; class C : public A {};
   B b;
   C *c = (C*)&b;
   ```

7. Use `static_cast` and `dynamic_cast` (**no** C-style casts) to demonstrate the following casts:

   (a) Cast `double a` to an integer.

   (b) Cast `Animal *a` to a `Cat*` (from Homework 11).

   (c) Cast `Rational r` to a `string`.