ECE326 – Fall 2019: Week 7 Exercise Questions

1. True or False [1 mark each]

Circle T is true, otherwise circle F for false.

1. A covariant tuple parameter is always type-safe. **(T)** **F**

   E.g. Tuple<Apple> can always replace Tuple<Fruit> (tuples are immutable).

2. Widening conversion guarantees you can get back the original data if needed. **T** **(F)**

   Loss of precision can occur.

3. In C++, type inference for variable declaration (using the auto keyword) cannot fail. **(T)** **F**

4. Macro systems do not have knowledge of the underlying programming language. **(T)** **F**

5. `explicit` keyword is used to prevent implicit conversion of class objects when the class overloads the cast operator. **T** **(F)**

   It's used to prevent implicit construction of an object that fits an argument and its constructor fits the input argument to a function, e.g. calling foo(1) for foo(Bar x) and Bar has a constructor Bar(int i).

2. Short Answers

1. Give an example to show that contravariant function return type is not type safe. **[2 marks]**

```
class Animal {}
class Cat : public Animal {}

struct CatShelter {
     virtual Cat * adopt();
};

struct AnimalShelter : public CatShelter {
     virtual Animal * adopt();
};

CatShelter * cs = new AnimalShelter();
Cat * cat = cs->adopt(); // type-unsafe, Animal may not be a Cat
```

2. What is the value of the following expression? What is the name of the behaviour? Assume the integer is 32-bit. **[2 marks]**

```
int a = 1 << 31;
printf("%d", -a);
```

```
-2147483648 (a == -a)
Overflow
```

3. Write a safe C macro, CIRCLE_AREA, that takes one parameter and will calculate the radius of a circle. You also need to define the constant PI. **[3 marks]**

```
#define PI 3.14159
#define CIRCLE_AREA(r) PI*(r)*(r)
/* better yet (GNU C extension) */
#define CIRCLE_AREA(r) ({ typeof(r) _r = (r); PI*_r*_r; })
```

4. Write the precondition and postcondition for the square root function, sqrt() that takes in a double and returns a double. Suppose Complex is a subclass of double, what's the variance relationship between double sqrt(double) and Complex sqrt(Complex)? **[3 marks]**

Let n be the parameter to sqrt, i.e. sqrt(double n), and r be the return value

Precondition: n >= 0

Postcondition: n >= 0 (if only the principle square root is returned)

Covariant parameter, covariant return type.

Note that both parameter and return type are problematic because covariant parameter is not type safe, and the postcondition of sqrt() is weakened, which violates Liskov's substitution principle.

3. Tagged Union **[10 marks]**

Complete the implementation of the tagged union example from class by adding a copy constructor, overload the equality operator, and type-safe getter/setter function

```
struct Tagged {
     enum { INT, STR } tag;
     union {              // anonymous union (members
          int * i;        // can enter parent scope)
          string * s;
     };
     Tagged(int i) : tag(INT), i(new int(i)) {}
     Tagged(const char * s) : tag(STR), s(new string(s)) {}
     ~Tagged() {
          if (tag == INT) delete i; else delete s;
     }

     Tagged(const Tagged & rhs) {

     }

     bool operator==( const Tagged & rhs) {


     }

     const string * get_str() const {


     }

     void set_str(const string & s) {



     }
};
```

see solution file (union.cpp)