This contains practice problems for lvalue and rvalue questions, since these are not covered on previous exams.

```
/X.h
class X { };

// main.cpp
void foo(X& i) {
   std::cout << "foo(X& i)" << std::endl;
}

void foo(const X& i) {
   std::cout << "foo(const X& i)" << std::endl;
}

void foo(X&& i) {
   std::cout << "foo(X&& i)" << std::endl;
}
int main(int argc, char** args) {
   X x;
   X& xr = x;
   const X xc;
   const X& xrc = xc;

   foo(x);    // Q1
   foo(xr);   // Q2
   foo(xc);   // Q3
              // Q4 is foo(xc) and lvalue or rvalue?
   foo(xrc);  // Q5
   foo(X( )); // Q6

   X xq = X( ); // Q7 is xq and lvallue or rvalue?
                // Q8 is X( ) and lvallue or rvalue?
}
Output:
foo(X& i)
foo(X& i)
foo(const X& i)
foo(const X& i)
foo(X&& i)
```

Q1: foo(X& i) is called because x is an lvalue and C++ converts lvalues into references. it is not const, and thus is passed to the non-const function rather than f(const X& i).

Q2: foo(X& i) is called because xr is an lvalue reference. It is not const, and thus is passed to the non-const function rather than f(const X& i).

Q3: foo(const X& i) is called because xc is both a const and an lvalue. C++ uses const and volatile to decide what function to call.

Q4: foo(xc) is an rvalue, what it returns does not have an identifiable memory location. In general, unless a function returns an lvalue reference (i.e., X&) a function call is an rvalue.

Q5: foo(const X& i) is called because xrc is both a const and an lvalue reference.

Q6: foo(X&& i) is called because X( ) is an rvalue, and must be passed to a foo that takes an X rvalue.

Q7: xq is an lvalue, because it has an identifiable memory location, i.e., the memory that the variables xq is in.

Q8: X( ) creates a temporary, the temporary has no identifiable memory location, and therefore X( ) is an rvalue.