

Function Arguments and Move Semantics Solutions

Constructor Argument

- What is the most efficient way of initializing the `m_str` member of this class, using both lvalues and rvalues?

```
class Test {  
    string m_str;  
public:  
    ...  
};
```

// Public member functions

- Explain your answer

Constructor Argument

- The most efficient way to initialize m_str is to write two constructors
- The const string& version is called when an lvalue is passed
- This results in one copy constructor call
- The string&& version is called when an rvalue is passed
- This results in one move constructor call

```
class Test {  
    string m_str;  
public:  
    Test::Test(const string& str) : m_str(str) { }           // Called when lvalue passed  
    Test::Test(string&& str) : m_str(std::move(str)) { }     // Called when rvalue passed  
    ... // Public member functions  
};
```

Constructor Argument

- The other options are:
- Pass by value/pass by move
`Test::Test(string str) : m_str(str) { }`
- Requires 2 copy constructor calls for lvalue, 1 copy + 1 move for rvalue
- Pass by value, then move
`Test::Test(string str) : m_str(std::move(str)) { }`
- Requires 1 copy + 1 move calls for lvalue, 2 moves for rvalue
- Pass by rvalue reference does not work with lvalues