Conversions & Const-ref

Unlike value-type variables, references have no implicit conversions. For instance, the following compiles and runs, because even though **a** is type **int** and **b** is type **double**, the compiler will **implicitly** create a temporary **int** value to "stand in" for **b**.

```
int a = 42;
double b = a;     // implicitly double b = int(a)
```

The following code, however, **will not** compile, because **x** is an **int**, but **rx** is a **reference to** a **double**. If **rx** were a **double**, (as in the previous example), instead of a **double&** then **x** **would be** promoted and stored in **rx**.

```
int x = 3;
double& rx = x;     // ILLEGAL. x is not a double
```

Constant References

While regular references must refer to an ***lvalue*** of exactly the same type, a **constant reference** may refer to a **literal** or **temporary** value. Here are some examples:

```
int& lit = 3;               // ILLEGAL
const int& lit2 = 3;        // OK, const-ref
string& str = "OK";         // ILLEGAL
const string& str2 = "OK";  // OK
```