Chapter 8 Review Questions

1. Short functions (that typically fit on one line) are good candidates for inline status.

2. a. `void song (const char * name, int times = 1);`

   b. `void song (const char * name, int times);`

   c. `void song (const char * name = "O, My Papa", int times = 1);`
   Note that you can't add a default value to a particular argument unless you also use default values for all arguments to its right.

3. a. `void iquote(int foo) { std::cout << "\"" foo << "\""; }`
   b. `void iquote(double foo) { std::cout << "\"" foo << "\""; }`
   c. `void iquote(std::string foo) { std::cout << "\"" foo << "\"";}`

4. a.
```
void display (const box & foo)
{
   std::cout << foo.maker << foo.height << foo.width <<
        foo.length << foo.volume;
}
```

   b.
```
void setvolume (box & foo)
{
   foo.volume = (foo.width * foo.height * foo.length);
}
```

5.
```
void fill (std::array<double, Seasons> & pa);
void show (const std::array<double, Seasons> & da);

fill (std::array<double, Seasons> & pa)
{
   for (int i = 0; I < Seasons; i++)
   {
        cout << "Enter " << Snames[i] << " expenses: ";
        cin >> &pa[i];
   }
}

void show (std::array<double, Seasons> & da)
{
} // no changes to function body
```

6. a. Can be accomplished by default arguments.

```
double mass (double density, double volume = 1);
```

Can also be accomplished by function overloading.

```
double mass (double density, double volume);
double mass (double density);
```

b. Can only be accomplished by function overloading since all arguments to the right of a default argument also must be default.

```
void repeat (int n, std::string text);
void repeat (std::string text);
```

c. Can only be accomplished by function overloading because default arguments cannot vary their type

```
int average (int num1, int num2);
double average (double num1, double num2);
```

d. This can't be done.

7.
```
template <typename anyType>
anyType returnLarger (anyType a, anyType b)
{
   if (a > b)
        return a;
   else
        return b;
}
```

8.
```
template <typename anyType>
anyType returnLarger (anyType a, anyType b)
{
   if (a.volume > b.volume)
        return a;
   else
        return b;
}
```

9. v1: float
v2: & float
v3: & float
v4: int
v5: double