C++ Primer Plus, 5th *Edition* by Stephen Prata
Chapter 9: Memory Models and Namespaces
Review Questions

1. What storage scheme would you use for the following situations?

   a. `homer` is a formal argument (parameter) to a function.

      Automatic.

   b. The `secret` variable is to be shared by two files.

      Global (static with external linkage).

   c. The `topsecret` variable is to be shared by the functions in one file but hidden from other files.

      Static with internal linkage.

   d. `beencalled` keeps track of how many times the function containing it has been called.

      Static with no linkage.

2. Describe the differences between a `using` declaration and a `using` directive.

   The `using` *declaration makes a single function, structure, constant, or member of the namespace available throughout the scope of the declaration. The* `using` *directive makes every function, structure, constant, and member of the namespace available throughout the scope of the declaration.*

3. Rewrite the following so that it doesn't use `using` declarations or `using` directives:

```
#include<iostream>
using namespace std;
int main()
{
    double x;
    cout << "Enter value: ";
    while (! (cin >> x))
    {
        cout << "Bad input. Please enter a number: ";
        cin.clear();
        while (cin.get() != '\n')
            continue;
    }
    cout << "Value = " << x << endl;
    return 0;
}
```

See the following code:

```
#include<iostream>
int main()
{
    double x;
    std::cout << "Enter value: ";
    while (! (std::cin >> x))
    {
        std::cout << "Bad input. Please enter a number: ";
        std::cin.clear();
        while (std::cin.get() != '\n')
            continue;
    }
    std::cout << "Value = " << x << std::endl;
    return 0;
}
```

4. Rewrite the following so that it uses `using` declarations instead of the `using` directive:

```
#include<iostream>
using namespace std;
int main()
{
    double x;
    cout << "Enter value: ";
    while (! (cin >> x))
    {
        cout << "Bad input. Please enter a number: ";
        cin.clear();
        while (cin.get() != '\n')
            continue;
    }
    cout << "Value = " << x << endl;
    return 0;
}
```

See the following code:

```
#include<iostream>
int main()
{
    using std::cout;
    using std::cin;
    using std::endl;
    double x;
    cout << "Enter value: ";
    while (! (cin >> x))
    {
        cout << "Bad input. Please enter a number: ";
        cin.clear();
        while (cin.get() != '\n')
            continue;
    }
    cout << "Value = " << x << endl;
    return 0;
}
```

5. Say that the `average(3,6)` function returns an `int` average of the two `int` arguments when it is called in one file, and it returns a `double` average of the two `int` arguments when it is called in a second file in the same program. How could you set this up?

   We could create a function called `average()` *in a given namespace which accepts two* `int`*s and returns an* `double`*. Then we could create a program were we declare and define the function* `average()` *which accepts two* `int`*s and returns a* `double`*. If we wanted to use the function that returned an* `int`*, we would call* `average()`*. If we wanted to use the function that returned a double, we would use* `<namespace>::average()` *where* `<namespace>` *represents the namespace we created for the function.*

6. What will the following two-file program display?

```
// file1.cpp
#include<iostream>
using namespace std;
void other();
void another();
int x = 10;
int y;

int main()
{
    cout << x << endl;
    {
        int x = 4;
        cout << x << endl;
        cout << y << endl;
    }
    other();
    another();
    return 0;
}

void other()
{
    int y = 1;
    cout << "Other: " << x << ", " << y << endl;
}

// file2.cpp
#include<iostream>
using namespace std;
extern int x;
namespace
{
    int y = -4;
}

void another()
{
    cout << "another (): " << x << ", " << y << endl;
}
```

It would print the following:

```
10
4
(garbage)
Other: 10, 1
another (): 10, -4
```

7. What will the following program display?

```
#include <iostream>
using namespace std;
void other();
namespace n1
{
    int x = 1;
}

namespace n2
{
    int x = 2;
}


int main()
{
    using namespace n1;
    cout << x << endl;
    {
        int x = 4;
        cout << x << ", " << n1::x << ", " << n2::x << endl;
    }
    using n2::x;
    cout << x << endl;

    other();
    return 0;
}

void other()
{
    using namespace n2;
    cout << x << endl;
    {
        int x = 4;
        cout << x << ", " << n1::x << ", " << n2::x << endl;
    }
    using n2::x;
    cout << x << endl;
}
```

We would see the following:

```
1
4, 1, 2
2
2
4, 1, 2
2
```