Stephen Tambussi
4/27

1) Inserts the new item at the beginning of the sequence
2)

```
#include <iostream>
using namespace std;
class box
{
    private:
        static int count;
        double length;
        double breadth;
        double height;
    public:
        box(double l=2.0,double b=2.0,double h=2.0)
        {
            count++;
            length = l;
            breadth = b;
            height = h;
            cout << "Number of box objects created so far:
" << count << endl;
        }
        double volume()
        {
            return length*breadth*height;
        }
};
int box::count;
int main(void)
{
    box Box1(3.3, 1.2, 1.5);
    box Box2(8.5, 6.0, 2.0);
    return 0;
}
```

3)

```
#include <iostream>
class small {
    public:
        small()
        {
            size = 0;
        };
        void k() const;
```

```
        void h(int i);
        friend void f(small z);
    private:
        int size;
};
void small::k() const
{
    small x, y;
    x = y; //ILLEGAL
    x.size = y.size; //ILLEGAL
    x.size = 3; //ILLEGAL
};
void small::h(int i)
{

};
void f(small z)
{
    small x, y;
    x = y; //LEGAL
    x.size= y.size; //LEGAL
    x.size = 3; //LEGAL
    x.h(42); //LEGAL
};
int main()
{
    small x, y;
    x = y; //LEGAL
    //x.size = y.size; //ILLEGAL
    //x.size = 3; //ILLEGAL
    x.h(42); //LEGAL
    return 0;
}
```

4)

```
class fruit {
    private:
        static int weight;
        static int color;
};
int fruit::weight = 1;
int fruit::color = 2;
int main() {
    fruit *fruit_ptr;
    fruit_ptr = new fruit[100];
}
```

5) Heap variables are essentially global in scope because they reside in a region of memory that is not managed automatically, unlike the stack.

```
int main()
{
   int *ptr  = new int[10];
}
```

6) No it is not possible to use the 'this' keyword inside a friend function. This is because a friend function is not within the scope of the class that it is "friends" with and the 'this' keyword is used to refer to a current instance of an object in the scope of the class.
7) The code does compile and run.
   Output: `Employee::foo()Computer::process()`