ITEM 6

Date:.....

Disallow compilers from generating functions you don't want.

Eg: Homesale h1;
Homesale h2;

Homesale h3(h1); // Avoid
h1=h2; // Avoid

In the above case, even if we do not declare copy constructor and copy assignment operator, compiler will make it by itself.

If we need to avoid it, we can declare the functions as private.

```
Class Homesale
{ public:
    ==
  private:
    Homesale (const Homesale&);
    Homesale& operator=(const Homesale&);
};
```

Function parameters are not given since we are never going to define it since we need to avoid copy.

Date:.....

The technique is not foolproof since member & friends funct^n can still call private funct^n but they will get linker error.

To move linker error to compile error make a class Uncopyable which is base class for HomeSale, its copy ctor & copy assignment are private.

```
class Uncopyable
{ protected:
      Uncopyable() {}
     ~Uncopyable() {}
  private:
    Uncopyable(const Uncopyable&);
    Uncopyable& operator=(const Uncopyable&);
};

Class HomeSale: private Uncopyable {
};
```

Thus class HomeSale no longer declare copy ctor & operator. & the member & friend funct^n call will be rejected at compile time.