Date:.....

Item 23 : Prefer non-member non-friend functions to member functions.

Consider a class WebBrowser.

```
Class WebBrowser {
    public:
    ...
        void clearCache();
        void clearHistory();
        void removeCookies();
    ...
    };
```

To perform all these actn together, WebBrowser might have funcn

```
Class webBrowser {
    public:
        void clearEverything();
    ...
    };
```

This functionality can be provided by non-member funcn also :

```
void clearBrowser(WebBrowser& wb)
{   wb.clearCache();
    wb.clearHistory();
    wb.clearCookies();
}
```

Date:.....

I Which one is better → member funcn or non-member funcn.

→ Object oriented principles dictate data & functions that operate on them should be bundled together

→ This is incorrect in this sense since object oriented dictat that data should be encapsulated.

→ If something is encapsulated, it is hidden from view, thus lesser chances of changes affecting directly and greater flexibility.

→ Only members and friends have access to private data members and non member & non-friend don't have access to any, thus providing greater encapsulation since it does not increase # the number of funcn that can access private parts of class.

II ~~non~~ that non-member funcn can be a funcn of another class. till that class is not a part of (or friend of) that class, it does not affect encapsulation.

Date:.....

```
namespace webBrowserStuff
{ class webBrowser { ... };
  void clearBrowser(webBrowser & wb);
}
```

We can put the non member non friend funcn in same namespace to that of class. We can have many convinience functions like this. We can store them in different <.h> files.

```
// header "webBrowser.h"
namespace webBrowserStuff
{
    class webBrowser { ... };
    ...
}
// header "webBrowserBookmark.h"
namespace webBrowserStuff
{ ....      // bookmark related funcn.
}
// header "webBrowserCookies.h"
namespace webBrowserStuff
{ // cookie & related funcn.
}
```

Date:.....

→ This is same way we use different .h in C++ like <vector> <list> we use only that .h which is needed for our operation, each is declared in the namespace std.

→ Instead of having single <C++ Standard lib> above approach is better.

→ Putting all convinience funcn in multiple header file - but one namespace also means that client can easily extend the set of convinience funcn

→ All they have to do is to add more non member non friend funcn in the namespace.

→ Above functionality cannot be given by classes, since we can derive them but still cannot access private data members & all classes may not be base class.