CMSC 341

Defensive Programming Tactics
Review of Tactics

- Well commented code is not only easier to read & debug, but also leads to more secure code

```cpp
void List<Object>::
insert(const Object &x, const ListItr<Object> &p)
{
    if (!p.isPastEnd()) // check to make sure we don’t
    { // overwrite beyond where we
        // intend in memory
        p.current->next =
            new ListNode<Object>(x,p.current->next);
    }
}
```

- Make defensive programming statements in your code similar to loop invariants
I/O Validation

• Always check the input and output of a program to make sure, the data your program is interacting with is valid
  – Make sure input/output buffers have a fixed maximum length, and you respect that length!
  – For Example verify on input that if you’re expecting integer data you’re getting integer data
I/O Validation Example

```c
int main ()
{
    string str;
    cout << "Please enter your name\n";
    getline(cin,str);
    cout << "Hello," << str << "!\n";
}
```

- If you don’t check the length of `str` you could be printing pages of data instead of the 20 or so character name you were expecting
  - Better implementation would be:

```c
if (str.size() < MAX_INPUT_SIZE)
    cout << "Hello," << str << "!\n";
```
Defend your Objects!

• Keep private data and methods private!
  – Prevents data from being overwritten maliciously or by accident

• Provide public methods and iterators that check for odd values before accessing private data

• When destroying objects with a destructor ensure that sensitive data is zeroed out before it’s released
Insecure Functions

• Wherever possible avoid using old insecure C functions

- `strcpy(3), strcat(3), sprintf(3), and gets(3)`

- Replace with: `strncpy(3), strcat(3), snprintf(3), and fgets(3)` // each checks for length during operation

- Avoid `strlen(3)` unless you’re first checked to make sure the string is terminated by a null
Post Processing of Code

• Run a security bug finder after you’ve finished writing your code. Then do what it says!
  – ITS4 is a command line tool that can be integrated into emacs and MS visual studio see http://www.cigital.com/its4/ for more info and to download
  – /GS a new part of MS Visual C++ .NET 2003 will automatically check for buffer-overflows