CSE 403
Lecture 17

Coding

Step through your code
- Maguire
  - Step through new code in the debugger the first time it is used
    - Add code, set break points, run debugger
    - Add code, run tests, if failure, run debugger
- Knuth
  - Developed tool to print out first two executions of every line of code

Candy machine interfaces
- Error prone return values or arguments
  ```c
  char c;
  c = getchar();
  if (c == EOF) ...  
  ```
- Classic bad example, getchar() returns an int!
- Alternate approach
  ```c
  bool fGetChar(char pch);
  ```
- Many bugs with malloc returning NULL

Another coding quiz
```c
char tolower(char ch) {
}
```  

Handling out of range inputs
- Ignore
- Return error code
- Assert
- Redefine the function to do something reasonable
- Write functions that, given valid inputs, cannot fail

Code tuning
- Don't overestimate cost
- A true story
  ```c
  A heated debate on whether to code a key function in Excel in assembly or C
  ```
  ```c
  The difference was 12 cycles
  ```
  ```c
  Eventually an engineer instrumented the three hour Excel torture test, and found that this function was called 76,000 times
  ```
  ```c
  The net savings was . . .
Efficiency gains
- Algorithmic improvement
  - $O(n \log n)$ sorting vs. $O(n^2)$ sorting
- Resource usage
  - File write vs. Memory write
- Inner loop of key routine
  - [Knuth] 4% of code uses 50% of runtime

Efficiency
- Avoid recomputation
- Cache results
  ```c
  int foo(int n)
  { 
    if (n < 0 || n >= fooCache.Length)
      throw new Exception("Foo err");
    if (inCache[n])
      return fooCache[n];
    else {
      fooCache[n] = computerFoo(n);
      inCache[n] = true;
      return fooCache[n];
    }
  }
  ```

Efficiency
- Reductions in strength
  - if ($\sqrt{x} < \sqrt{y}$)...

Efficiency vs. Clarity
- Almost always favor clarity over efficiency
  - The human reader is more important
  - Compilers and processors have improved significantly in the last four decades

Efficiency
- Memory management
  - Even in Java / C#
    - Preallocating/reusing large objects
    - Buffer pool

Debugging
- What are the key steps in debugging a program?
Kernigan and Pike's debugging wisdom

- Look for common patterns
  - Common bugs have distinct signatures
  - int n; scanf("%d", &n);
- Examine most recent change
- Don't make the same mistake twice
- Debug it now, not later
- Get a stack trace
- Read before typing

K & P, II

- Explain your code to someone else
- Make the bug reproducible
- Divide and conquer
  - Find simplest failing case
- Display output to localize your search
  - Debugging with printf()
- Write self checking code
- Keep records