CS 261 – Data Structures

Preconditions, Postconditions & Assert
Preconditions

preconditions are input conditions for the function

1. magic is only required to do its task if pre-conditions are satisfied
2. The caller knows that *if he satisfies* those conditions, magic will perform the task correctly

```c
/*
pre: size < SIZELIMIT
pre: name != null;
post: result >= MINRESULT
*/
int magic (int size, char *name)
{
    assert(size < SIZELIMIT);
    assert(name != null);
    ... DO STUFF ...
    assert(result >= MINRESULT);
    return result;
}
```
Postconditions

Postconditions are output conditions for a function

1. magic guarantees the condition will hold when it completes. As developer, you must ensure this!
2. The caller is certain of what it will get, provided it has met preconditions

```c
/*
  pre: size < SIZELIMIT
  pre: name != null;
  post: result >= MINRESULT
*/

int magic (int size, char *name)
{
  assert(size < SIZELIMIT);
  assert(name != null);
  ... DO STUFF ...
  assert(result >= MINRESULT);
  return result;
}
```
Pre-conditions + Post-conditions

When combined....they define a contract!!!

Using pre and post conditions and CHECKING them helps you find and remove bugs and fulfill the contract!
In practice....

- put pre-conditions in the header
- put post-conditions in the header
- during debugging, use `asserts()` to enforce them

To catch errors when recovery is generally not immediately possible

Useful during debugging, but we should replace for “Release”
Bugs and Errors

1. Program Error: a bug, and should never occur

   Replace these asserts with a reasonable error message

2. Run-time Error: can validly occur at any time during execution (e.g. user input is illegal) and should be recoverable

   Write recovery code for these kinds of errors