Automated Unit Testing

EECS 168 Programming 1
David Janzen
Perfection is Impossible

• No one writes perfect code the first time, every time
• How do we find out if code is correct?
  – Testing
• Forms of testing
  – **Compiling** tests for valid syntax
  – **Acceptance testing** involves running the program as a user testing for correct operation
  – **Unit testing** involves testing individual units (functions)
• Once we know there are defects, we must fix them through code review and debugging
Exhaustive Testing is Impossible

• Even a simple one-parameter function can have an infinite number of inputs

```c
float square(float number)
{
    return number * number;
}
```

• So we test with a representative set of input/output combinations

-14.8, -1, 0, 0.0001, 5.987, 22, 1025.9
Manual Unit Testing

- Individual units can be tested by writing drivers

```cpp
// This function sums the integers from min to max inclusive.
// Pre: min < max
// Post: return-value = min + (min+1) + ... + (max-1) + max
int sum(int min, int max);

int main()
{
    int first, second;
    cout << "Enter two integers, the first smaller" << endl;
    cin >> first >> second;
    cout << "sum(" << first << "," << second << ") is " << sum(first, second) << endl;
    return 0;
}
```
Automatic Unit Testing

- Individual units can be tested by writing automated tests with assert()
- assert() takes one boolean parameter

```cpp
#include <cassert>
int sum(int min, int max);

int main()
{
    assert(sum(0,2) == 3);
    assert(sum(-2,2) == 0);
    assert(sum(3,7) == 25);
    return 0;
}
```
assert()  
- If assert’s parameter is false, then program execution is halted and a message is given.

```cpp
#include <cassert>
int sum(int min, int max);

int main()
{
  assert(sum(3,7) == 15);
  return 0;
}

$./a.out
assertion "sum(3,7) == 15" failed: file "sumtest.cpp", line 36
```
Assert() alternative

- The following[1] gives extra information

```c
#include <cassert>
#define Assert(b,s) { if (!(b)) s; assert(b); }
int sum(int min, int max);

int main()
{
    Assert(sum(3,7) == 15,
           cout << "sum(3,7) = " << sum(3,7) << endl);
    return 0;
}
```

1. Contributed by Dr. John Gauch
Organizing Tests

```cpp
#include <cassert>
void run_tests();
int sum(int min, int max);

int main()
{
    run_tests();
    //what the program actually does
    return 0;
}

void run_tests()
{
    assert(sum(0,2) == 3);
    assert(sum(-2,2) == 0);
    assert(sum(3,7) == 25);
}
```
int sum(int min, int max)
{
    int sum=0;
    for(int i=min; i<=max; i++)
    {
        sum += i;
    }
    return sum;
}

int sumrec(int min, int max)
{
    if(min == max)
    {
        return min;
    }
    if(min < max)
    {
        return min + sumrec(min+1,max);
    }
}
void run_tests()
{
    assert(sum(3, 7) == 25);
    assert(sumrec(3, 7) == 25);
    assert(sum(-2, 3) == 3);
    assert(sumrec(-2, 3) == 3);
    assert(sum(-5, 5) == 0);
    assert(sumrec(-5, 5) == 0);
}
When do we test?

• Traditional linear/waterfall model
• “Big design up-front”
Test-First vs. Test-Last

- Test-Last process
  1. Design software
  2. Write code
  3. Write unit tests
  4. Repeat to 2

- Test-First process
  1. Write a unit test
  2. Write code to make test pass
  3. Refactor code and test
  4. Repeat to 1