CSc 352: Testing and Code Coverage
Testing and test cases

```c
int main()
{
    read x;
    if (x is odd) {
        compute payroll data;
    }
    else {
        delete all files;
        send rude email to boss;
        crash computer;
    }
}
```

- make sure you use at least fifty different test inputs
- 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ...
Testing and test cases

```c
int main()
{
    read x;
    if (x is odd) {
        compute payroll data;
    }
    else {
        delete all files;
        send rude email to boss;
        crash computer;
    }
}
```

make sure you use at least fifty different test inputs

1, 3, 5, 7, 9, 11, 13, 15, 17, 19, ...

It isn’t enough to have a lot of test cases. We have to make sure our tests “cover” the program adequately.
gcov: a code coverage analyzer

• test coverage program
  – indicates how many times each line was executed
  – marks code that did not get executed
  – cumulative over a set of tests

  helps you understand
  – how effectively your current test cases “cover” the code
  – what additional test inputs you need in order to get better coverage

• needs the program to be compiled with additional gcc options
An example

```c
#include <stdio.h>

int fib(int n) {
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    }
    else {
        int i, f, f0 = 0, f1 = 1;
        for (i = 1; i < n; i++) {
            f = f0 + f1;
            f0 = f1;
            f1 = f;
        }
        return f;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    printf("fib(%d) is %d\n", n, fib(n));
    return 0;
}
```

additional compiler flags

```
% gcc -Wall -fprofile-arcs -ftest-coverage fib.c
% ./a.out
2
fib(2) is 1
% ./a.out
3
fib(3) is 2
% ./a.out
4
fib(4) is 3
% ./a.out
5
fib(5) is 5
% ./a.out
6
fib(6) is 8
% gcov fib.c
File 'fib.c'
Lines executed:86.67% of 15
```

testing didn’t execute all of the code
An example

```c
#include <stdio.h>

int fib(int n) {
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    } else {
        int i, f, f0 = 0, f1 = 1;
        for (i = 1; i < n; i++) {
            f = f0 + f1;
            f0 = f1;
            f1 = f;
        }
        return f;
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("fib(%d) is %d\n", n, fib(n));
    return 0;
}
```
An example

```c
#include <stdio.h>

int fib(int n)
{
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    }
    else {
        int i, f, f0 = 0, f1 = 1;
        for (i = 1; i < n; i++) {
            f = f0+f1;
            f0 = f1;
            f1 = f;
        }
        return f;
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("fib(%d) is %d\n", n, fib(n));
    return 0;
}
```
An example

```c
#include <stdio.h>

int fib(int n)
{
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    }
    else {
        int i, f, f0 = 0, f1 = 1;
        for (i = 1; i < n; i++) {
            f = f0+f1;
            f0 = f1;
            f1 = f;
        }
        return f;
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("fib(%d) is %d\n", n, fib(n));
    return 0;
}
```
#include <stdio.h>

int fib(int n)
{
    if (n <= 0) {
        return 0;
    }
    else if (n == 1) {
        return 1;
    }
    else {
        int i, f, f0 = 0, f1 = 1;
        for (i = 1; i < n; i++) {
            f = f0+f1;
            f0 = f1;
            f1 = f;
        }
        return f;
    }
}

int main()
{
    int n;
    scanf("%d", &n);
    printf("fib(%d) is %d\n", n, fib(n));
    return 0;
}
Code coverage and testing

• Just because every line has been executed does not mean the program has been tested thoroughly
  – we may want to test the same line of code under different conditions
    • e.g.: a loop should be tested with values that cause 0, 1, and “many” iterations

• However, if some lines are not executed the program is definitely not thoroughly tested
  – gcov helps us identify and fix this
    • exception: “system errors” that may be difficult to create
Example of not enough testing

This is (almost) the program we wrote in class to convert all lower case letters in a string to upper case.
Example of not enough testing

Compile and test it. 100% of code is executed and the result is correct. So the code has no bugs, right?

```c
#include <stdio.h>

void myup(char c[]) {
    int i;

    for (i=0; c[i] != '\0'; ++i) {
        if (c[i] > 'a' && c[i] < 'z')
            c[i] = c[i] - 'a' + 'A';
    }
    return;
}

int main () {
    char st[64];

    scanf("%63s",st);
    myup(st);
    printf("%s\n", st);
    return 0;
}
```

$gcc -fprofile-arcs -ftest-coverage -Wall -o upper upper.c
$upper
ThisTest1.
THISTEST1.
$gcov upper.c
File 'upper.c'
Lines executed:100.00% of 10
upper.c:creating 'upper.c.gcov'
Example of not enough testing

Compile and test it. 100% of code is executed and the result is correct. So the code has no bugs, right?

There were still errors.
gcov: summary

• code coverage testing tool
  – works with gcc; needs additional compiler flags
    • gcc –fprofile-arcs –ftest-coverage ...

• shows execution frequency of each line of code
  – reports % of lines covered by tests
    • coverage values are cumulative
    • delete *.gcda file to start afresh
  – how many times each line was executed
  – highlights lines not executed