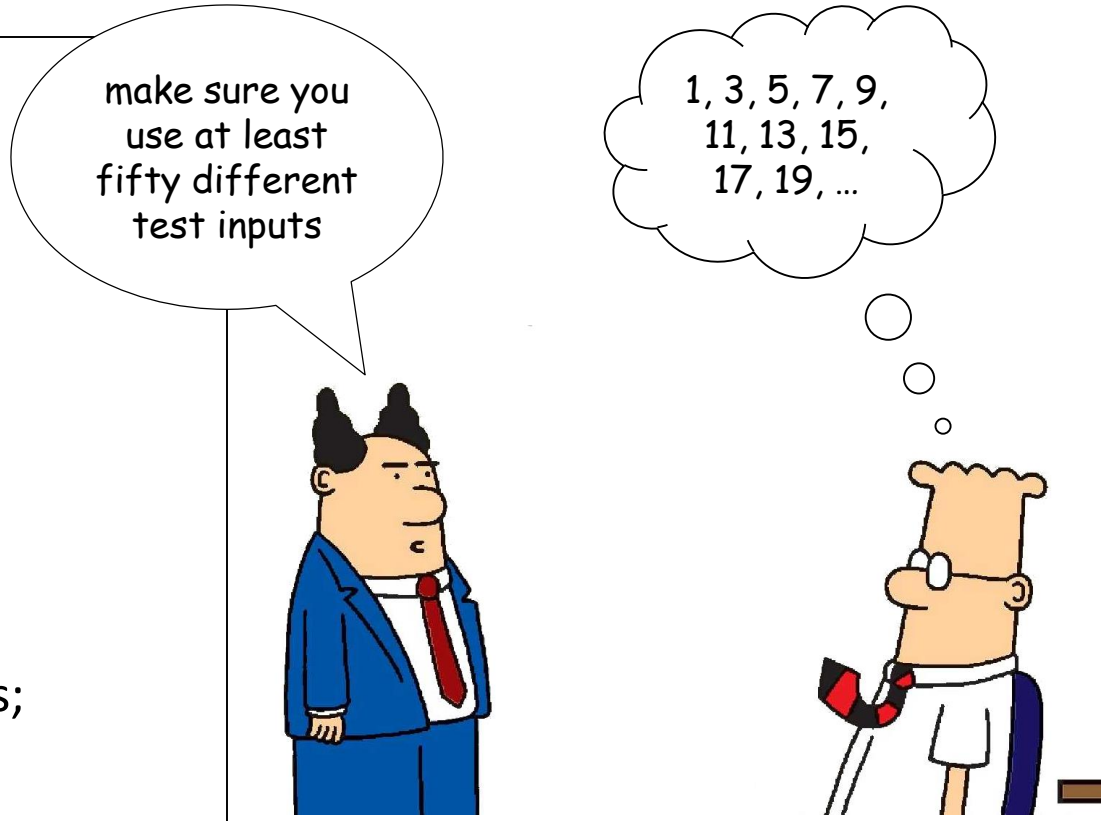
CSc 352: Testing and Code Coverage

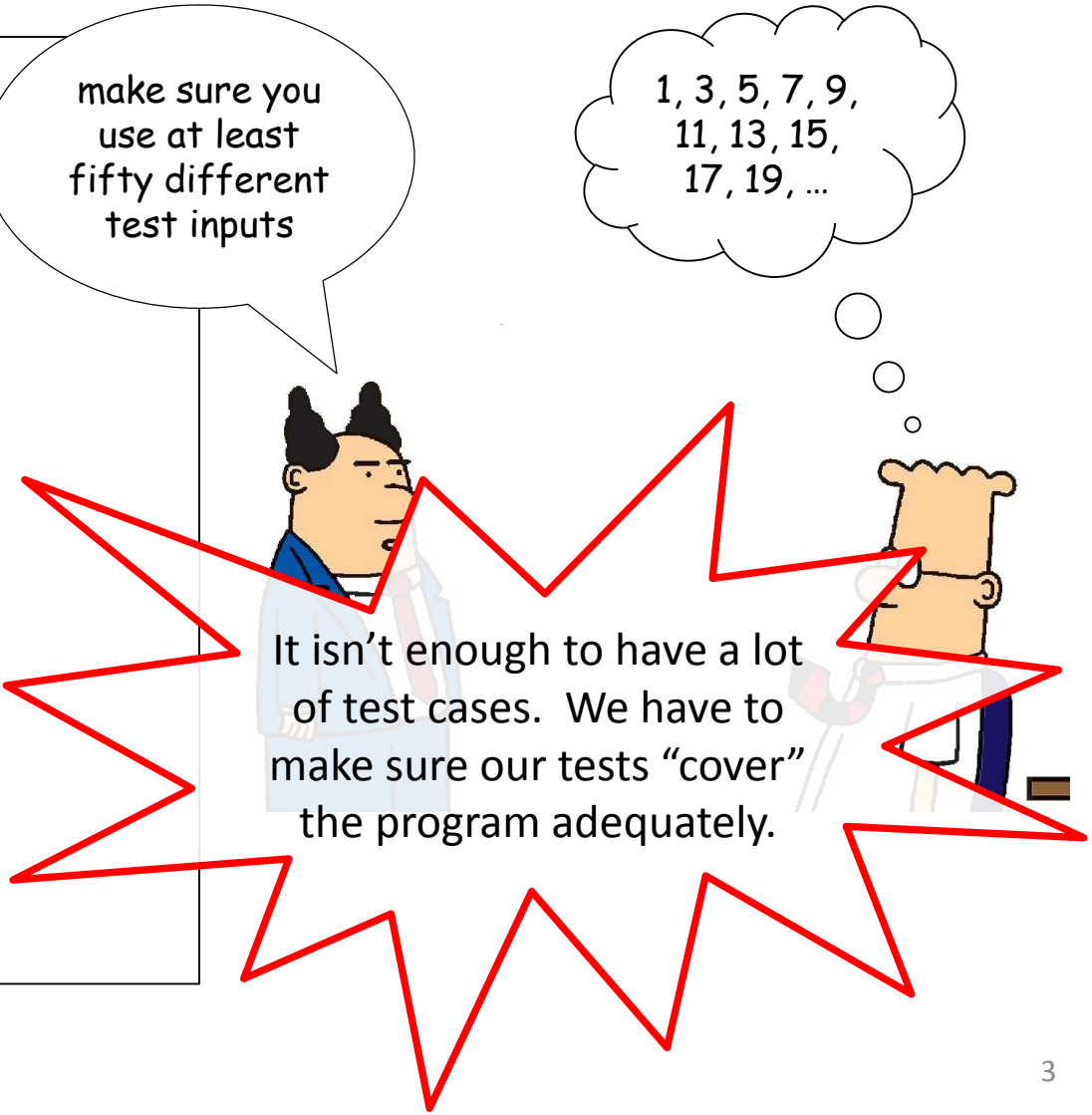
Testing and test cases

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



Testing and test cases

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



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
```

input

input

input

input

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;
}
```

input

```
% gc
% ./
2
fib(
% ./
3
fib(
% ./
4
fib(
% ./
5
fib(
% ./
6
fib(
% gc
File
Line
```

```
% more fib.c.gcov
        -:    0:Source:fib.c
        -:    0:Graph:fib.gcno
        -:    0:Data:fib.gcda
        -:    0:Runs:1
        -:    0:Programs:1
        -:    1:
        -:    2:
        -:    3:#include <stdio.h>
        -:    4:
        1:    5:  int fib(int n)
        -:    6:  {
        1:    7:    if (n <= 0) {
#####:    8:      return 0;
        -:    9:    }
        1:   10:    else if (n == 1) {
#####:   11:      return 1;
        -:   12:    }
        -:   13:    else {
        1:   14:      int i, f, f0 = 0, f1 = 1;
        -:   15:
        8:   16:      for (i = 1; i < n; i++) {
        7:   17:        f = f0+f1;
        7:   18:        f0 = f1;
        7:   19:        f1 = f;
        -:   20:      }
        -:   21:
        1:   22:      return f;
        -:   23:    }
        -:   24:  }
        -:   25:
        1:   26:  int main()
        -:   27:  {
        -:   28:    int n;
        -:   29:
        1:   30:    scanf("%d", &n);
        -:   31:
        1:   32:    printf("fib(%d) is %d\n", n, fib(n));
        -:   33:
        1:   34:    return 0;
        -:   35:  }
```

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;
}
```

input

```
% ./a.out
1
fib(1) is 1
% gcov fib.c
fib.c:source file is newer than notes file 'fib.gcno'
(the message is only displayed one per source file)
File 'fib.c'
Lines executed:93.33% of 15
Creating 'fib.c.gcov'
```

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;
}
```

input

```
% ./a.o
1
fib(1)
% gcov
fib.c:s
(the me
File 'f
Lines e
Creatin
```

```
% more fib.c.gcov
        -:    0:Source:fib.c
        -:    0:Graph:fib.gcno
        -:    0:Data:fib.gcda
        -:    0:Runs:2
        -:    0:Programs:1
        -:    0:Source is newer than graph
        -:    1:
        -:    2:
        -:    3:  #include <stdio.h>
        -:    4:
        2:    5:  int fib(int n)
        -:    6:  {
        2:    7:    if (n <= 0) {
    #####:    8:      return 0;
        -:    9:    }
        2:   10:    else if (n == 1) {
        1:   11:      return 1;
        -:   12:    }
        -:   13:    else {
        1:   14:      int i, f, f0 = 0, f1 = 1;
        -:   15:
        8:   16:      for (i = 1; i < n; i++) {
        7:   17:        f = f0+f1;
        7:   18:        f0 = f1;
        7:   19:        f1 = f;
        -:   20:      }
        -:   21:
        1:   22:      return f;
        -:   23:    }
        -:   24:  }
        -:   25:
        2:   26:  int main()
        -:   27:  {
        -:   28:    int n;
        -:   29:
        2:   30:    scanf("%d", &n);
        -:   31:
        2:   32:    printf("fib(%d) is %d\n", n, fib(n));
        -:   33:
        2:   34:    return 0;
        -:   35:  }
```

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;
}
```

input

```
% ./a.out
0
fib(0) is 0
% gcov fib.c
fib.c:source file is newer than notes file 'fib.gcno'
(the message is only displayed one per source file)
File 'fib.c'
Lines executed:100.00% of 15
Creating 'fib.c.gcov'
```

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

```
eanson@lectura:~/oldHome/inClass$ cat upper.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;
}
eanson@lectura:~/oldHome/inClass$
```

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

```
eanson@lectura:~/oldHome/inClass$ cat upper.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;
}
```

```
eanson@
$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'
```

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

Example of not enough testing

```
eanson@lectura:~/oldHome/inClass$ cat upper.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;
}
eanson@lectura:~/oldHome/inClass$
```

There were still errors.

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

```
$upper
San_Diego_zoo
SaN_DIEGO_zOO
$
```

Oops!

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