

1. Rewrite assert.cpp to using assert() instead of the if it currently uses (2 points).

```
struct Point
{
    double x;
    double y;
};

#include <cassert>
#include <iostream>
using namespace std;

int main()
{
    double dx, dy;
    Point p1, p2;

    /* get points */
    cout << "First point (x, y): ";
    cin >> p1.x >> p1.y;
    cout << "Second point (x, y): ";
    cin >> p2.x >> p2.y;

    /* earn points by finding the slope */
    dx = p2.x - p1.x;
    dy = p2.y - p1.y;
    assert(dx != 0);
    cout << "The slope is " << dy / dx << endl;
    return 0;
}
```

2. Compile & run overStep.cpp with the gdb flag turned on as per the instructions on the web site. What is the output from gdb after it receives the segmentation fault (2 points)?

```
Program received signal SIGSEGV, Segmentation fault.
0x0804850d in main () at overStep.cpp:8
8          numbers[i] = i;
```

3. Set a breakpoint in `infiniteFunctions.cpp` at the beginning of `f()`. Step about 5 – 10 times and then do a backtrace. What are the results of the backtrace (2 points)?

```
#0 g (x=9) at infiniteFunctions.cpp:20
#1 0x08048593 in f (x=8) at infiniteFunctions.cpp:14
#2 0x080485ab in g (x=7) at infiniteFunctions.cpp:20
#3 0x08048593 in f (x=6) at infiniteFunctions.cpp:14
#4 0x080485ab in g (x=5) at infiniteFunctions.cpp:20
#5 0x08048593 in f (x=4) at infiniteFunctions.cpp:14
#6 0x080485ab in g (x=3) at infiniteFunctions.cpp:20
#7 0x08048593 in f (x=2) at infiniteFunctions.cpp:14
#8 0x080485ab in g (x=1) at infiniteFunctions.cpp:20
#9 0x08048593 in f (x=0) at infiniteFunctions.cpp:14
#10 0x08048576 in main () at infiniteFunctions.cpp:7
```

4. Set a breakpoint at lines 15 and 22 in `print.cpp` and continue a few times while saying yes to the “More Fibonacci” question. Print the values of  $i$ ,  $F(i)$ , and `keepGoing`. What is the output? Now say no the “More Fibonacci” question. Print the values of  $i$ ,  $F(i)$ , and `keepGoing`. What is the output at the second breakpoint (4 points)?

While saying “yes”, we get:

```
(gdb) print i
$1 = 5
(gdb) print F(i)
$2 = 5
(gdb) print keepGoing
$3 = true
```

And after saying “no”, we get:

```
(gdb) print i
$4 = 6
(gdb) print F(i)
$5 = 8
(gdb) print keepGoing
$6 = false
```

5. Sketch the graphical output (or paste a screenshot) of ddd with a breakpoint at line 39 of lab07.cpp and local arguments displayed (2 extra credit points).

