Unit Testing as Hypothesis Testing
Example Code

Jonathan Clark

September 19, 2012

evil_sort_test.c0

```
#use <conio>

#use "evil_sort.c0"

bool arrays_equal(int[] expected, int[] actual, int n) {
  bool result = true;
  for (int i=0; i<n; i++) {
    print("i=");
    printint(i);
    print(": expected ");
    printint(expected[i]);
    print(", actual ");
    printint(actual[i]);
    if (actual[i] != expected[i]) {
      result = false;
      print("*");
    }
    print("\n");
  }
  return result;
}

void test_empty() {
  print("test empty\n");
  int[] input = alloc_array(int, 0);
  int[] expected = alloc_array(int, 0);

  int[] result = evil_sort(input, 0);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 1));
```

```
}

void test_singleton() {
  print("test singleton\n");
  int[] input = alloc_array(int, 1);
  int[] expected = alloc_array(int, 1);

  int[] result = evil_sort(input, 1);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 1));
}

void test_123() {
  print("test 123\n");
  int[] input = alloc_array(int, 3);
  input[0] = 3;
  input[1] = 2;
  input[2] = 1;

  int[] expected = alloc_array(int, 3);
  expected[0] = 1;
  expected[1] = 2;
  expected[2] = 3;

  int[] result = evil_sort(input, 3);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 3));
}

void test_012() {
  print("test 012\n");
  int[] input = alloc_array(int, 3);
  input[0] = 2;
  input[1] = 0;
  input[2] = 1;

  int[] expected = alloc_array(int, 3);
  expected[0] = 0;
  expected[1] = 1;
  expected[2] = 2;

  int[] result = evil_sort(input, 3);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 3));
}
```

```
void test_negated() {
  print("test negated\n");
  int[] input = alloc_array(int, 3);
  input[0] = -2;
  input[1] = -1;
  input[2] = -3;

  int[] expected = alloc_array(int, 3);
  expected[0] = -3;
  expected[1] = -2;
  expected[2] = -1;

  int[] result = evil_sort(input, 3);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 3));
}

void test_already_sorted() {
  print("test already sorted\n");
  int[] input = alloc_array(int, 3);
  input[0] = 1;
  input[1] = 2;
  input[2] = 3;

  int[] expected = alloc_array(int, 3);
  expected[0] = 1;
  expected[1] = 2;
  expected[2] = 3;

  int[] result = evil_sort(input, 3);
  //@assert( \length(result) == \length(expected) );
  assert(arrays_equal(expected, result, 3));
}

int main() {
  test_singleton();
  //test_empty();
  test_123();
  //test_012();
  //test_already_sorted();
  //test_negated();
  return 0;
}
```

evil_comparator_test.c0

```
#use <conio>

#use "evil_sort.c0"

int main() {
  assert(is_leq(0,0) == true);
  assert(is_leq(-3,-2) == true);
  assert(is_leq(-2,-3) == false);
  return 0;
}
```

safe_add_test.c0

```
#use <conio>

#use "safe_add.c0"

int main() {
  assert(safe_add(0,0) == 0);
  assert(safe_add(-1,1) == 0);
  assert(safe_add(1,-1) == 0);
  return 0;
}
```

safe_add_fail.coin

```
safe_add(2147483647, 1);
safe_add(-2147483648, -1);
```

test.sh

```
cc0 -dx evil_sort_test.c0
cc0 -dx evil_comparator_test.c0
cc0 -dx safe_add_test.c0
coin -d safe_add.c0 < safe_add_fail.coin
```