CSC108H Lecture 22

Dan Zingaro

November 2, 2012

Choosing Test Cases

- We generally cannot call a function with all possible inputs
  - e.g. if a function takes an integer as a parameter, there are an infinite number of values with which the function can be called
- Luckily, the number of tests we run is not as important as choosing good test cases
- We can divide all possible inputs into meaningful categories
- Then, we choose a representative test from each category

ConcepTest

```
def our_max(num1, num2):
  '''(number, number) -> number
  Return the larger of num1 and num2.
  '''
```

Of the following sets of test cases, which one is best?

- A. (2, 3), (2, 2), (-4, -5), (1.2, 1.2), (1.2, 1.5)
- B. (2, 3), (2, 2), (4, 4), (6, 6), (-4, -5), (1.2, 1.2), (1.2, 1.5)
- C. (2, 3), (2, 2), (3, 2), (-4, -5), (1.2, 1.2), (1.2, 1.5)
- D. (2, 3), (3, 2), (2, 2)

ConcepTest

```
def insert_after(L, n1, n2):
  '''(list of int, int, int) -> NoneType
  Insert n2 after each occurrence of n1 in L.
  '''
```

Start from the top and consider each test in order. Vote for the letter of the first test that adds nothing to what was covered by the previous tests.

- A. ([2, 4, 6], 3, 8)
- B. ([2, 4, 6], 4, 8)
- C. ([2, 4, 2], 2, 8)
- D. ([2, 4, 4, 2], 2, 8)
- E. ([2, 2], 2, 8)

ConcepTest

```
def insert_after(L, n1, n2):
  '''(list of int, int, int) -> NoneType
  Insert n2 after each occurrence of n1 in L.
  '''
```

Start from the top and consider each test in order. Vote for the letter of the first test that adds nothing to what was covered by the previous tests.

- A. ([1], 3, 8)
- B. ([5], 3, 8)
- C. ([5, 6, 7], 5, 8)
- D. ([5, 6, 7], 2, 8)

Example: Dictionary Function

Provide a set of test cases for this function.

```
def inc_count(d, k):
  '''(dict of {object:int}, object) -> NoneType
  k is immutable. Increment the value associated with k
  in d. If k is not a key in d, add k with value 1.
  '''
```

ConcepTest

```
def indices(big, small):
  '''(str, str) -> list of int
  Return the indices of big at which non-overlapping
  copies of small start. small is non-empty.

  >>> indices('A Coool pool look', 'oo')
  [3, 9, 14]
  '''
```

Of the following sets of test cases, which one is best?

- A. `('', 'a')`, `('cool pool', 'oo')`, `('cool pool', 'pp')`
- B. `('', '')`, `('cool pool', 'oo')`, `('cool pool', 'pp')`
- C. `('cool pool', 'oo')`, `('cool pool', 'pp')`
- D. `('', 'a')`, `('', '')`, `('cool pool', 'oo')`, `('cool pool', 'pp')`

Example: Indices

For which tests does this bad implementation fail? How can it be fixed?

```
def indices(big, small):
  '''(str, str) -> list of int
  Return the indices of big at which
  non-overlapping copies of small start.

  >>> indices('A Coool pool look', 'oo')
  [3, 9, 14]
  '''
  index = 0
  indices = []
  while index != -1:
    indices.append(big.find(small, index))
    index = big.find(small, index + len(small))
  return indices
```