AMACSS Seminar

Term Test 1 Review

Possible topics on the term test

- ADT's
  - Designing an ADT
  - Coding an ADT from a docstring
  - Writing a docstring for an ADT
- Containers
  - Manipulating containers to achieve a certain goal
- Linked Lists
  - Operations on linked lists
- Blackbox testing
  - Testing an ADT without knowing how it's implemented

Making a stack out of queues, and vice versa

1 Stack = 2 Queues

How? Well let's take a look…

Try to make a Queue out of stacks

Try the following question

Question 1. [10 MARKS]

Write the body of the function below so that it satisfies its docstring. Assume that module `stack.py` defines a class `Stack` that provides the usual methods: `is_empty()`, `push(item)`, `pop()`.

For full marks, your code must *not* depend on any details of the implementation of class `Stack`. In other words, the only thing you can do with a `Stack` object is to call some of its methods.

```
from stack import Stack

def size(stk):
    """(Stack) -> int
    Return the number of items on Stack stk, *without* modifying stk.
    (It's OK if the contents of stk are modified during the execution of this
    function, as long as everything is restored before the function returns.)
    """
    # Hint: You can use more than one stack.
```

Try the following question

Write the body of the function below so that it satisfies its docstring.

```
def reverse(head):
    """(LLNode) -> LLNode

    Given the head pointer to a linked list,
    reverse the linked list and return a pointer
    to the head of the reversed list.)
    """
```

Write testcases for your banana_verify function from exercise 2. These testcases don't need to be written in unittest format, but the assertions should be written in proper assertion format. i.e assertEqual, etc.

Your testcases should include examples where:

- To many letters in source or goal
- More gets than puts
- Too many moves
- Too little moves