Question 1. [10 MARKS]

Implement a class that models a cash register in a store. This cash register will know what the HST tax rate is (charged on all sales, for simplicity), is able to make sales, and keeps track of cash received as sales and tax. Your class implementation should include only the following (these are the only parts we will grade):

- a declaration of class name, and a class docstring
- an _**init**_ method
- a method to make a sale for a given price and amount of cash paid by the customer, recording the money paid (including tax, which this method calculates), and returning the amount of change owed
- a method to report the total number of sales made and the total cash received (including tax)
- an _**eq**_ method to report whether the attributes of one cash register are equivalent to those of another cash register

All methods must have proper docstrings, except no examples are required.

Question 2. [10 MARKS]

Implement a class that models a quiz question. A quiz question provides the question text, and a user is able to enter a response to that text. Once a response is entered, a quiz question reports whether the response is correct or not, by comparing it to the correct answer.

Also implement two subclasses to model multiple-choice quiz questions, and numerical quiz questions. Multiple choice quiz questions accept responses that are one of: "a", "b", "c", "d", or "e", and the correct answer must be one of these. Numerical quiz questions accept responses that are floats, and a correct answer is one that is in a given range, for example (0.99, 1.01).

Your design of these classes should aim to minimize duplicate code, except that **all** methods that are defined in the subclasses should also be defined in the superclass (although perhaps not implemented). You should write docstrings for each class and method.

Indicate which methods are inherited, overridden, or extended, with a brief comment explaining why you chose each approach (inherited, overridden, or extended) for these two subclasses.

For this question, we do **not** require _str_ or _eq_ methods.

Question 3. [8 MARKS]

Read over the definition of **count_stack** below, then complete its implementation. Your function implementation may create as many extra instances of class Stack as you like (**hint**: this is a good idea), but the **only** methods of Stack you may use are:

add(obj) add **obj** to the top of this Stack

remove() remove and return top element of this Stack

is_empty() return whether this Stack is empty

You **may not** use any Python lists, tuples, dictionaries, or other sequence classes. You may create variables to represent ordinary Python objects, such as ints.

```
def count_stack(s):
    """
    Return the number of elements in Stack s.  Restore
    s to the same state it started in.

    @param Stack s:
```

```
    @rtype: int

    >>> s1 = Stack()
    >>> s1.add("how")
    >>> s1.add("now")
    >>> count_stack(s1)
    2
    """
```