PLEASE HAND IN

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

Term test #1

CSC 148H1, Section L5101/L5102
Duration — 50 minutes

Student Number: _____

Last Name: _____

First Name: _____

Lecture Room (circle one): HS610 PB B150

Do **not** *turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

This test consists of 3 questions on 8 pages (including this one). *When you receive the signal to start, please make sure that your copy of the test is complete.*

Please answer questions in the space provided. You will earn 20% for any question you leave blank or write "I cannot answer this question," on. You will earn substantial part marks for writing docstrings, even if the implementation is missing.

Good Luck!

Question 1. [10 MARKS]

Implement a class that models a bank account. This account will know how to deposit money, withdraw money, and report the current balance in the account. Your class implementation need include only the following (the only parts we will grade):

- a declaration of class name, and a class docstring
- an **__init__** method that sets up the account with an initial balance
- a method to make a deposit
- a method to make a withdrawal, charging a $10 overdraft **and** refusing the withdrawal if there are not enough funds
- a method to report the current balance

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

This page is left (mainly) blank for things that don't fit elsewhere.

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.

This page is left (mainly) blank for things that don't fit elsewhere.

Question 3. [8 MARKS]

Read over the docstring of **bottom_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 bottom_stack(s):
    """
    Return the bottom element of Stack s, or None if s
    is empty.  Restore s to the same state it started in.

    @param Stack s: Stack to get to the bottom of
    @rtype: object|None

    >>> s1 = Stack()
    >>> s1.add("one")
    >>> s1.add("two")
    >>> bottom_stack(s1)
    'one'
    """
```

This page is left (mainly) blank for things that don't fit elsewhere.

This page is left (mainly) blank for things that don't fit elsewhere.

1: _____/10

2: _____/10

3: _____/ 8

TOTAL: _____/28