Computer Science 134C **Student Number:**

Midterm Examination, Spring 2018 **Preferred Superpower:**

Instructions: This is a 50-minute closed book examination. All work should be your own. Concise and elegant answers will receive the best score.

Question	Points	Score
0.	15	
1.	25	
2.	15	
3.	25	
4.	20	
Total:	100	

Question 0. Marathon times are measured with second accuracy.

a. Write a function that converts a floating point number of seconds to whole seconds according to 2018-19 IAAF Rule 165.10.b:

> "For races partially or entirely outside the stadium, unless the time is an exact whole second, the time shall be converted and recorded to the next longer whole second, e.g. 2:09:44.3 shall be recorded as 2:09:45."

```
def convert(elapsed):
    """Converts a floating point elapsed time to the an integer number
       of seconds, rounding up to the next second if there is a non-zero
       fractional part.  Meets IAAF 165.10.b specifications.
       >>> convert(0.0)
       0
    """
```

b. The doc-comment test is not really convincing. For example, the one test does not check against rounding downward. Write two more doc-comment tests that help to verify `convert`.

Question 1. To help with anonymous grading, I maintain a list of integer ID numbers for our class, `cs134IDs`. Write the following procedures to help with managing these ID values. Feel free to call any of these procedures in any of the other definitions.

a. Actually, I really only look at the rightmost three digits, which uniquely identify members of our class. Write a function, `last(n,id)`, that returns the last `n` digits of an `id` as an integer. (Hint: `str`.)

```
    def last(n,id):
        """Returns an integer that represents the last n digits of id.
           >>> last(2,3030618)
           18
           >>> last(4,3030618)
           618
        """
```

b. Write a function that returns a sorted version of `idList` based on just the last `n` digits of each ID.

```
    def sortedIDs(n, idList):
        """Returns idList sorted based on the last n digits.  idList remains unchanged.
           >>> sortedIDs(3, cs134clist)
           [3026032, 3035056, 3024075, ..., 3030993]
        """
```

c. Write a function that returns True if the last n digits of integers in idList are unique to each ID.

```
def unique(n, idList):
    """Returns True if the last n digits of the IDs in idList are unique.
       >>> unique(0,[])
       True
       >>> unique(2,cs134IDs)
       False
       >>> unique(3,cs134IDs)
       True
    """
```

Question 2. The following code shuffles a list by randomly selecting elements from pile and appending them to result. It does not currently work. Fix 3 logical errors while maintaining the spirit of the code.

```
from random import randint
def shuffle(pile):
    """Returns a new list of the elements of pile in random order. """
    n = len(pile)    # size of pile
    result = None    # the list of shuffled elements
    copied = set()   # indices of elements of pile in result
    while len(copied) <= n:
        # pick a random item to copy to result
        location = randint(0,n-1)
        while location in copied:
            location = randint(0,n-1)
            copied.add(location)
        result += pile[location]
    return result
```

Question 3. Rewrite each of these Python snippets in a more elegant manner.

a.
```
result = b == False  # assume b is a boolean value
```

b.
```
result = True if (a or b) else False # assume that a and b are booleans
```

c.
```
if vowel == 0:    # assume vowel is an integer from 0 to 4.
    c = 'a'
elif vowel == 1:
    c = 'e'
elif vowel == 2:
    c = 'i'
elif vowel == 3:
    c = 'o'
elif vowel == 4:
    c = 'u'
```

d.
```
if c == 'a':     # assume c is a lowercase vowel: a, e, i, o, or u.
    vowel = 0
elif c == 'e':
    vowel = 1
elif c == 'i':
    vowel = 2
elif c == 'o':
    vowel = 3
elif c == 'u':
    vowel = 4
```

e.
```
l = len(s)        # assume s is a string
allSame = True
for i in range(l):
   for j in range(l):
       if s[i] != s[j]:
           allSame = False
```

Question 4. Answer the following with reasonably short, concise answers.

a. How does `__all__` support module abstraction?

b. Write a list comprehension that creates a list, `sq`, of squares of integers 1 to 1000, inclusive.

c. Suppose we create a file `README` in a directory managed by `git`. What three steps are needed to send the file to the `git` server?

d. What does the following recursive procedure do?

```
def zip(s):
    if len(s) < 2:
        return True
    else:
        return (s[0] == s[-1]) and zip(s[1:-1])
```

I have neither given nor received aid on this exam. Initialed:

⋆