15-112 F21 Practice Quiz 8

25 minutes

Note: This practice quiz was written by TAs. It may not reflect the exact difficulty or length of the actual quiz.

1. **Short Answer & Multiple Choice**

Multiple Choice: Which of the following is not an O(1) operation?

A. Popping the last element from a list
B. Making a list exactly 100 elements long
C. Finding the largest integer in a set
D. Getting the length of a dictionary

Multiple Choice: Which of the following is true?

A. Looping over all the elements in a set takes O(1) work.
B. It is possible to write a comparison-based sorting algorithm that runs in O(n).
C. We can search a list in O(log n) only if that list is already in sorted order.
D. Values in a dictionary are subject to the same restrictions as elements in a set.

Short Answer: Given the list [2, 5, 1, 3], what are the first four **comparisons** made by selectionSort, as it works in xSortLab (which selects the maximum value on each pass)?

Multiple Choice: Consider the following code.

```
class Dog(object):
    def __init__(self, name):
        self.name = name
class Chihuahua(Dog): pass
B = Dog("Buddy")
W = Chihuahua("Wink")
```

Which of the following is true?

A. `Dog` and `Chihuahua` have the same constructor
B. `type(W) == Dog` evaluates to `True`
C. `str(W)` returns the string `'Chihuahua("Wink")'`
D. `isinstance(B, Chihuahua)` evaluates to `True`

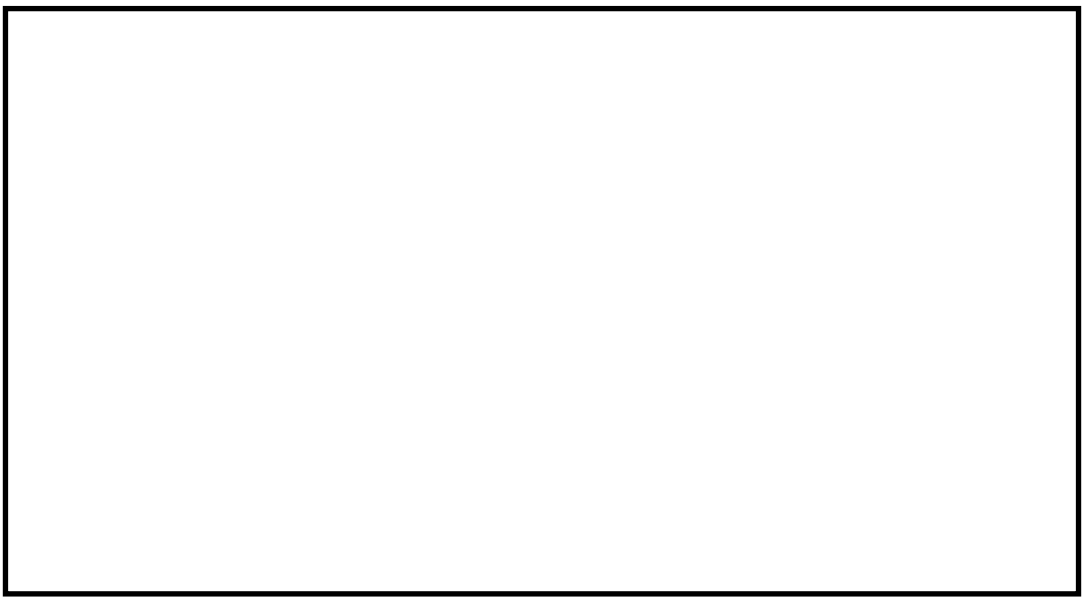
Short Answer: If a function with O(log n) work takes 5 seconds to run on an input of size 10, approximately how many seconds would it take to run on an input of size 1000? You may assume the log is base 10.

2. **Code Tracing**: Indicate what these print. Place your answers (and nothing else) in the boxes below the code.

```
import copy
def ct1(n):
    w = set([])
    b = {1, 9}
    while n > 0:
        w.add(n % 10)
        b.add(n % 100 // 10 + 2)
        n //= 100
    l, o = w, copy.copy(b)
    for item in b:
        if (item - 1) % 5 != 4:
            o.remove(item)
        else:
            for i in range(-4, 3, 2):
                l.add(item + i)
    for quiz in (b, o, w, l):
        print(sorted(quiz))
print(ct1(37135))
```

```
import copy
def ct2(d):
    a = copy.copy(d)
    b = copy.deepcopy(d)
    for n in [2, 2, 1, 5]:
        if n in d:
            if isinstance(d[n], list):
                d[n].append(b[2*n])
            else:
                d[n] = a.get(n, "quiz")
        else:
            d[n] = 'come' * n
            a[0] += 1
            y = 0
            for k in b:
                y = max(k, y)
            b[2 * y] = '!'
    for swag in (d, a, b):
        print(swag) # you may write the keys in any order
    return [d[1], a[0], d[2]] + a[5]

d = {0: 0, 5: ['bowl']}
print(ct2(d))
```



3. **Free Response:** `mostPopularFriend(d)`

Recall that `friendsOfFriends(d)` takes a dictionary `d` of the following form:

```
d = dict()
d["fred"] = set(["wilma", "betty", "barney"])
d["wilma"] = set(["fred", "betty", "dino"])
```

With this in mind, write the function `mostPopularFriend(d)` that takes a dictionary of that form, and returns the name that occurs the most number of times in all the sets of friends. In the example above, `mostPopularFriend(d)` would return `"betty"`. You may assume that there is exactly one most popular friend, so you can ignore ties.

4. **Free Response:** IceCream Class

Write the `IceCream` class such that the following test cases pass.

```
def testIceCreamClass():
    print("Testing Ice Cream Class ...", end = " ")
    c1 = IceCream(['Vanilla']) # Make a vanilla IceCream - yum!
    assert(c1.flavors == ['Vanilla'])
    assert(c1.scoops == 2) # IceCream objects always begin with 2 scoops

    c1.addFlavors("Strawberry") # We can add a flavor
    assert(c1.flavors == ['Vanilla', 'Strawberry'])
    c1.addFlavors("Strawberry") # but we can only add any given flavor once!
    assert(c1.flavors == ['Vanilla', 'Strawberry'])

    c1.addScoop() # add a scoop
    assert(c1.scoops == 3)

    c2 = IceCream(['Pepperoni'])
    c3 = IceCream(['Paprika', 'Pepperoni'])
    # We can combine two IceCream objects to make a new IceCream!
    c4 = c2.combine(c3)
    # The new IceCream has no duplicate flavors; we only keep the first
    # occurrence of a flavor (looking at c2's flavors list first)
    assert(c4.flavors == ['Pepperoni', 'Paprika'])
    assert(c4.scoops == 4) # sum of the original IceCream objects' scoops

    # What's the point in making ice cream if you can't eat it?
    c4.eat() # consume one scoop
    assert(c4.scoops == 3)
    for i in range(3): # eat up all the scoops
        assert(c4.eat() == None)
    # When all the scoops have been eaten, return a message
    assert(c4.eat() == 'no more ice cream')

    print("Passed!")
```

You may write your solution on this page.