CS 331 Spring 2019

Midterm Exam

Instructions:

- This exam is closed-book, closed-notes. Computers of any kind are not permitted.
- For numbered, multiple-choice questions, fill your answer in the corresponding row on the "bubble" sheet.
- For problems that require a written solution (labeled with the prefix "WP"), write your answer in the space provided on the written solution sheet. Please write legibly and clearly indicate your final answer.
- Turn in the exam question packet, bubble sheet, and written solution sheet separately.

Basic Concepts (24 points):

1. What are the contents of the list `lst` after the following code is executed?

```
lst = [x+y for x in ['a', 'the', 'one']
           for y in [' car', ' fish']]
```

   (a) `['a car', 'the car', 'one car', 'a fish', 'the fish', 'one fish']`
   (b) `['a car', 'a fish', 'the car', 'the fish', 'one car', 'one fish']`
   (c) `[['a car', 'a fish'], ['the car', 'the fish'], ['one car', 'one fish']]`
   (d) `['a car fish', 'the car fish', 'one car fish']`

2. Consider the following iterator implementation:

```
class MyIt:
    def __init__(self):
        self.x = 100

    def __iter__(self):
        return self

    def __next__(self):
        if self.x < 0:
            raise StopIteration
        else:
            self.x -= 20
            return self.x
```

   What is the result of the expression `[x for x in MyIt()]`?

   (a) `[100, 80, 60, 40, 20, 0]`
   (b) `[80, 60, 40, 20, 0, -20]`
   (c) `[80, 60, 40, 20, 0]`
   (d) An exception is raised before a result is computed.

3. Consider the following generator function:

```
def my_gen(i, j, k):
    while i <= j:
        if i % k == 0:
            yield k
        yield i
        i += 1
```

   What is the result of the expression `[x for x in my_gen(1, 8, 2)]`?

   (a) `[1, 2, 3, 2, 5, 2, 7, 2]`
   (b) `[1, 2, 3, 4, 5, 6, 7, 8]`
   (c) `[1, 2, 2, 3, 2, 4, 5, 2, 6, 7, 2, 8]`
   (d) `[2, 1, 2, 2, 2, 3, 2, 4, 2, 5, 2, 6, 2, 7, 2, 8]`

4. What is the worst-case runtime complexity of retrieving the last element given its index from an array-backed list of N elements?

   (a) $O(1)$
   (b) $O(\log N)$
   (c) $O(N)$
   (d) $O(N^2)$

5. What is the worst-case runtime complexity of swapping the values at two different indexes in an array-backed list of N elements?

   (a) $O(1)$
   (b) $O(\log N)$
   (c) $O(N)$
   (d) $O(N^2)$

6. Consider a scenario where we wish to search for an item in an unsorted array-backed list of N elements, but only care whether it appears in the second half of the list. What is the worst-case runtime complexity of this search?

   (a) $O(1)$
   (b) $O(\log N)$
   (c) $O(N)$
   (d) $O(N^2)$

7. What is the worst-case runtime complexity of using insertion sort to sort the contents of an array-backed list of N elements?

   (a) $O(1)$
   (b) $O(\log N)$
   (c) $O(N)$
   (d) $O(N^2)$

8. What is the maximum number of elements a properly implemented binary search will need to compare a value against in order to determine its position in a sorted list of 30,000 elements?

   (a) 5
   (b) 10
   (c) 15
   (d) 20

9. Which of the following relations is true?

   (a) $n! = O(n)$
   (b) $3n + 10 = O(n)$
   (c) $n^2 - 1000 = O(\log n)$
   (d) $2^n = O(n^2)$

10. What best describes the relationship between $f(n)$ and $g(n)$ if $f(n) = O(g(n))$?

    (a) as n gets large, $g(n)$ is less than or equal to some multiple of $f(n)$
    (b) as n gets large, $f(n)$ is less than or equal to some multiple of $g(n)$
    (c) there is some n that makes $f(n)$ less than $g(n)$
    (d) the maximum value of $f(n)$ is less than the maximum value of $g(n)$ (for positive n)

11. Which of the following datatypes in Python is mutable?

    (a) integer
    (b) string
    (c) tuple
    (d) dictionary

12. When might you prefer to use a list comprehension instead of a semantically equivalent generator expression to compute a sequence of values?

    (a) when we need to use the sequence as a target of a `for` loop
    (b) when the sequence will be iterated over once and then discarded
    (c) when the sequence will only be iterated over partially
    (d) when we need random access (by index) to the values in the sequence

Estimating Big-O (9 points):

For each of the following functions, determine the corresponding worst-case runtime complexity in terms of N. Assume that any `lst` arguments are Python lists.

13.
```
def fA(N):
    accum = 0
    for i in range(N):
        for j in range(i+1):
            accum += j
    return accum
```

(a) $O(1)$
(b) $O(\log N)$
(c) $O(N)$
(d) $O(N^2)$

14.
```
def fB(lst):
    N = len(lst)
    accum = 0
    for i in range(N):
        accum += lst[i]
    for j in range(N-1, -1, -1):
        accum += lst[j]
    return accum
```

(a) $O(1)$
(b) $O(\log N)$
(c) $O(N)$
(d) $O(N^2)$

15.
```
def fC(lst):
    N = len(lst)
    if N > 1:
        mid = N // 2
        if lst[mid] < lst[0] and lst[mid] < lst[N-1]:
            return lst[mid]
        elif lst[0] < lst[N-1]:
            return lst[0]
        else:
            return lst[N-1]
    else:
        return lst[0]
```

(a) $O(1)$
(b) $O(\log N)$
(c) $O(N)$
(d) $O(N^2)$

Lists and Dicts (6 points):

WP1 Implement `consolidate`, which accepts a list of lists in the parameter `lsts`, and returns a dictionary whose keys consist of values found in the input lists, which map to numbers indicating how many times each value appears across all the lists.

E.g., `consolidate([1,2,3], [1,1,1], [2,4], [1])` should return the dictionary `{1: 5, 2: 2, 3: 1, 4: 1}`.

Mystery Sort (8 points):

Consider the following mystery sort function:

```
def mystery_sort(lst):
    for i in range(len(lst), 0, -1):
        print(lst) # display list contents
        swapped = False
        for j in range(i-1):
            if lst[j] > lst[j+1]:
                lst[j], lst[j+1] = lst[j+1], lst[j]
                swapped = True
        if not swapped:
            break
```

WP2 (a) Show the list contents, in order, displayed by all calls to `print(lst)` when `mystery_sort` is called with the input list `[7, 5, 2, 8, 6, 3, 4, 1]`. (3 points)

WP2 (b) What is the Big-O runtime complexity of `mystery_sort`, when called with an input list of length N? (2 points)

WP2 (c) What sort of input list will result in the *best-case* runtime performance for `mystery_sort`? Explain. (3 points)

Array-backed List (8 points):

WP3 Implement the array-backed list method `insert_all`, which accepts a valid index `idx` and another list `other`, and inserts all values from `other` into the underlying array starting at index `idx`. Your implementation should not use any other array-list methods, and may only perform the following operations on the backing array (named `data` in the provided skeleton code):

- `len(self.data)`
- Accessing a valid, positive index (e.g., `self.data[i]`)
- `self.data.append(None)`
- `del self.data[len(self.data)-1]`

You may use `len(other)` to obtain the number of the elements in the other list, and access elements in `other` by valid, positive indexes (e.g., `other[i]`).

E.g., calling `insert_all(3, ['a', 'b', 'c'])` on a list with `data` contents `[0, 1, 2, 3, 4]` results in `data` being updated to `[0, 1, 2, 'a', 'b', 'c', 3, 4]`.

For full credit, your implementation should run in $O(M + N)$ time, where M is the number of elements in the array-backed list, and N is the number of elements in `other`.