CS 331 Fall 2017

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 = list(range(10))
lst[1:8] = [2*x+1 for x in range(1, 4)]
```

(a) `[0, 3, 5, 7, 9]`
(b) `[0, 3, 5, 7, 8, 9]`
(c) `[0, 3, 7, 15, 9]`
(d) `[0, 1, 1, 3, 5, 2, 3, 4, 5, 6, 7, 8, 9]`

2. What are the contents of the dictionary `dct` after the following code is executed?

```
dct = {}
for i in range(10, 0, -2):
    if i * 2 not in dct:
        dct[i] = i // 2
```

(a) `{2: 1, 6: 3, 8: 4, 10: 5}`
(b) `{2: 1, 4: 2, 6: 3, 8: 4, 10: 5}`
(c) `{20: 10, 16: 8, 12: 6}`
(d) `{1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 7: 14, 9: 18}`

3. What is the worst-case runtime complexity of locating and returning the last element in an unsorted array-backed list of N elements?

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

4. What is the worst-case runtime complexity of determining whether a given value exists in an unsorted array-backed list of N elements?

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

5. What is the worst-case runtime complexity of deleting a random element from an array-backed list of N elements?

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

6. Which of the following scenarios will consistently cause binary search (given search value x and list `lst`) to exhibit the poorest runtime complexity?

   (a) x is the least common value in `lst` (i.e., fewest duplicates)

   (b) `lst` contains duplicates of x

   (c) x is the middle element of `lst`

   (d) x is not found in `lst`

7. Consider the following function definition:

```
def gen():
    print(0)
    yield 10
    print(10)
    yield 20
```

   Which of the following assigns the value `10` to the variable `x`?

   (a)
   ```
   g = gen()
   x = next(g)
   ```

   (b)
   ```
   x = iter(gen())
   ```

   (c)
   ```
   _ = gen()
   x = gen()
   ```

   (d)
   ```
   g = gen()
   _ = next(g)
   g = next(g)
   ```

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 100,000 elements?

   (a) 8

   (b) 16

   (c) 24

   (d) 32

9. Which of the following relations is *not*, strictly speaking, true?

   (a) $3n + 2 = O(n)$

   (b) $2n^3 + 10n - 5 = O(n^3)$

   (c) $10^n - n^2 = O(n^2)$

   (d) $5\log_2 n = O(2^n)$

10. What do the variables `a` and `b` refer to, respectively, after the following code executes?

```
lst = 'red fish blue frog egg'.split()
it1 = iter(lst)
it2 = iter(lst)
next(it1), next(it2), next(it2)
a, b = next(it1), next(it2)
```

   (a) `frog` and `egg`

   (b) `blue` and `egg`

   (c) `red` and `frog`

   (d) `fish` and `blue`

11. Which of the following operations on some built-in Python list `lst` has $O(N)$ runtime complexity (assume that `i` and `j` are valid indices)?

   (a) `len(lst)`

   (b) `lst[i] = x`

   (c) `x = lst[j]`

   (d) `lst[i:j] = []`

12. Which of the following operations on some built-in Python list `lst` will *mutate* the list (assume that `i` and `j` are valid indices)?

   (a) `lst + lst`

   (b) `lst.extend(x)`

   (c) `lst.index(x, i, j)`

   (d) `lst * 7`

Estimating Big-O (9 points):

For each of the following functions, determine the corresponding worst-case runtime complexity when called with an input list of size N. Assume the input list is a Python (array-backed) list.

13.
```
def fA(N, x):
    accum = 0
    while N > 1:
        if N % x == 0:
            accum += N
        N = N - N/2
    return accum
```

(a) $O(1)$

(b) $O(\log N)$

(c) $O(N)$

(d) $O(N^2)$

14.
```
def fB(M, N):
    accum = 0
    for i in range(1, M, M//10):
        for j in range(1, N, N//10):
            if i < j:
                accum += i
            else:
                accum += j
    return accum
```

(a) $O(1)$

(b) $O(M)$

(c) $O(N)$

(d) $O(M \cdot N)$

15.
```
def fC(lst):
    N = len(lst)
    accum = 0
    if N < 100:
        return 0
    else:
        for i in range(N * 10):
            accum += i
        return accum
```

(a) $O(1)$

(b) $O(\log N)$

(c) $O(N)$

(d) $O(N^2)$

Lists and Dictionaries (6 points):

WP1 Complete the implementation of `max_repeat_counts`, which takes a non-empty list and returns a dictionary containing a key for each element in the list, with a value corresponding to the maximum number of times the element repeats (in succession).

E.g., `max_repeat_counts([1, 2, 2, 2, 2])` returns `{1: 1, 2: 4}`.

E.g., `max_repeat_counts([3, 3, 4, 4, 3, 4, 4, 4])` returns `{3: 2, 4: 3}`.

Insertion Sort (6 points):

Consider the following reversed insertion sort implementation which prints the contents of the list at the start of each inner iteration:

```
def rev_insertion_sort(lst):
    for i in range(1, len(lst)):
        for j in range(i, 0, -1):
            print(lst) # print list contents
            if lst[j] > lst[j-1]:
                lst[j-1], lst[j] = lst[j], lst[j-1]
            else:
                break
```

WP2 Show the list contents, in order, displayed by all calls to `print` when `rev_insertion_sort` is called with the input list `[2, 1, 3, 4, 5]`. The first output is already filled in for you; you may not need all lines.

Array-backed List (6 points):

WP3 Complete the implementation of the array-backed list method `remove_span` which should remove the first span of adjacent elements with the specified value from the list.

E.g., `remove_span(2)` on `[1, 1, 2, 2, 2, 3, 3, 2, 2]` results in `[1, 1, 3, 3, 2, 2]`.

E.g., `remove_span(5)` on `[3, 3, 4, 4, 5, 5, 5]` results in `[3, 3, 4, 4]`.

If the list does not contain the specified value, a `ValueError` should be raised.

Your implementation should assume elements are stored in a Python list referenced by `self.data`, which you can only manipulate as an array (using the rules given in class). You may not use any other ArrayList methods.