CS 331 Fall 2015

Midterm Exam 1

October 21^{st}, 2015

Happy Back to the Future Day!

Instructions:

- Please enter your selections in the provided bubble sheet (fill in the bubbles fully!)
- This exam is closed-book, closed-notes. Calculators are neither needed nor permitted.

Problem 1. (20 points):

Choose the best answer to each of the following questions.

1. What is the time-complexity of the following function?

```
def foo(n):
    res = 1
    for i in range(1, n):
        for j in range(1, n):
            res = i * j * res
    return res
```

   a. O(n)
   b. O(n^2)
   c. O(n^{10})
   d. O($\log n$)

2. What is the time-complexity of the following function?

```
def bar(n):
    res = 0
    while n > 0:
        res += n
        n = n // 2 # '//' is integer division
    return res
```

   a. O(n)
   b. O(n^2)
   c. O(n^{10})
   d. O($\log n$)

3. What is the time-complexity of the following function?

```
def baz(n):
    res = 0
    for val in range(1024 * n):
        res += val
    return res
```

   a. O(n)
   b. O(n^2)
   c. O(n^{10})
   d. O($\log n$)

Problem 2. (20 points):

Given the following variable definitions:

```
lst1 = ['x', 'y', 'z']
lst2 = ['<', '>']
nums     = (2, 3, 5, 7)
suffixes = ('ed', 'ish', 'e')
sentence = 'red fish yellow whale'
words    = sentence.split()
```

Choose the value of the object assigned to `x` at the end of each of the following snippets of code.

4. `x = [a+b+a for a in lst1 for b in lst2]`

   a. `['x<x', 'y<y', 'z<z', 'x>x', 'y>y', 'z>z']`
   b. `['x<y', 'y<z', 'x>y', 'y>z']`
   c. `['x<x', 'x>x', 'y<y', 'y>y', 'z<z', 'z>z']`
   d. `'x<xy<yz<zx>xy>yz>z'`

5. `x = [[b+a+b for a in lst1] for b in lst2]`

   a. `[['<x<', '<y<', '<z<'], ['>x>', '>y>', '>z>']]`
   b. `['<x<', '<y<', '<z<', '>x>', '>y>', '>z>']`
   c. `[['<x>', '<y>', '<z>'], ['>x<', '>y<', '>z<']]`
   d. `[['<x<', '>x>'], ['<y<', '>y>'], ['<z<', '>z>']]`

6.
```
lst3 = [lst2[i] + lst1[j] + lst2[k]
        for i in range(len(lst2))
        for j in range(len(lst1))
        for k in range(len(lst2))]
x = lst3[3]
```

   a. `'>x<'`
   b. `'<z<'`
   c. `'>y>'`
   d. `'<y>'`

7. `x = sum(i for i in range(10) if i not in nums)`

   a. `28`
   b. `0`
   c. `15`
   d. `45`

8. `x = sorted(words, key=lambda x: x[-1])`

   a. `['red', 'fish', 'yellow', 'whale']`
   b. `['red', 'whale', 'fish', 'yellow']`
   c. `['fish', 'red', 'whale', 'yellow']`
   d. `['red', 'yellow', 'whale', 'fish']`

9.
```
x = []
for w in words:
    x.extend([w for s in suffixes if w[1:] == s])
```

   a. `['red', 'fish', 'yellow', 'whale']`
   b. `['fish', 'whale']`
   c. `['yellow']`
   d. `['red', 'fish']`

10.
```
d = {}
for s in suffixes:
    for w in words:
        if s in d:
            d[s].append(w+s)
        else:
            d[s] = [w+s]
x = d['ish']
```

   a. `['redish', 'fishish', 'yellowish', 'whaleish']`
   b. `['fishish']`
   c. `['redish']`
   d. `[]`

11.
```
d = { c: [c+s for s in suffixes]
      for c in sentence }
x = d['w']
```

   a. `['whaleish']`
   b. `'wish'`
   c. `['wed', 'wish', 'we']`
   d. `None`

Problem 3. (20 points):

The following is a partial implementation of an array-backed list ADT. Read through the method docstrings, and select the answers that correctly implement the required functionality.

```
class ArrayList:
    def __init__(self):
        self.data = []

    def drop_first(self, n):
        """Removes the first `n` elements from the list.
        E.g., calling `drop_first(3)` on [5, 3, 1, 2, 4] gives [2, 4]"""
        if n > len(self.data):
            return
        for i in ____________:

            ____________
        for i in range(n):

            del self.data[_____________]
```

12. a. `range(n)`
    b. `range(1, n, 1)`
    c. `range(n, len(self.data))`
    d. `range(len(self.data)-1, n, -1)`

13. a. `self.data[i] = self.data[i+1]`
    b. `self.data[i] = self.data[i-1]`
    c. `self.data[i] = self.data[i+n]`
    d. `self.data[i-n] = self.data[i]`

14. a. `i`
    b. `n`
    c. `0`
    d. `len(self.data)-1`

```
def insert(self, idx, x):
    """Inserts the new element ‘x‘ at position ‘idx‘ in the list."""
    if idx > len(self.data):
        return
    self.data.append(None)

    for i in ____________:

        ------------
    self.data[idx] = x
```

15. a. `range(len(self.data)-1, idx, -1)`
    b. `range(len(self.data), 1, -1)`
    c. `range(len(self.data))`
    d. `range(idx)`

16. a. `self.data[i] = self.data[0]`
    b. `self.data[0] = self.data[i]`
    c. `self.data[i] = self.data[i-1]`
    d. `self.data[i+1] = self.data[i]`

```
def uniq(self):
    """Removes all repeated elements from the list except for their
    first instance. Assumes that __delitem__ has been implemented.
    E.g., calling ‘uniq‘ on [1, 2, 1, 3, 2, 1] gives [1, 2, 3]."""
    i = 0
    while i < len(self.data):
        x = self.data[i]

        j = ____________

        while _____________:

            if _____________:
                del self[j]
            else:
                j += 1
        i += 1
```

17. a. `0`
    b. `i + 1`
    c. `i - 1`
    d. `len(self.data)`

18. a. `j < i`
    b. `j >= 0`
    c. `j < len(self.data)`
    d. `j < len(self.data)-1`

19. a. `x == self.data[j]`
    b. `x in self.data[j]`
    c. `self.data[j-1] == self.data[j]`
    d. `self.data[j+i] == self.data[j]`

```
def sort(self):
    """Implements insertion sort (in ascending order) on the
    contents of the list."""
    for j in range(1, len(self.data)):
        to_insert = self.data[j]
        i = j - 1
        while ____________:

            ____________
            i -= 1

        ____________
```

20. a. `i >= 0 and self.data[i] > to_insert`
    b. `i > 0 and self.data[i] < to_insert`
    c. `i < len(self.data) and self.data[i] > to_insert`
    d. `i <= len(self.data)`

21. a. `self.data[i] = to_insert`
    b. `self.data[i+1] = self.data[0]`
    c. `self.data[i], self.data[j] = self.data[j], self.data[i]`
    d. `self.data[i+1] = self.data[i]`

22. a. `self.data.append(to_insert)`
    b. `self.data[j] = to_insert`
    c. `self.data[j+1] = to_insert`
    d. `self.data[i+1] = to_insert`

```
def scramble(self, n=100):
    """Swaps elements at two randomly generated positions ‘n‘ times."""
    for _ in range(n):
        i = random.randrange(len(self.data))
        j = random.randrange(len(self.data))

        ____________
```

23. a. `self.data[i] = self.data[j]`
    b. `self.data[j] = self.data[i]`
    c. `self.data[i], self.data[j] = self.data[j], self.data[i]`
    d. `self.data[i], self.data[j] = self.data[i], self.data[j]`