Quiz 0

August 31, 2010

lastname, firstname = _______________, _______________ auditing = ____________

Instructions
1. There are 100 points in this quiz, which is worth 5% of the course grade.
2. You have **only 20 minutes**. Time is very tight.
3. You can write only one expression or statement in one slot (______).
4. Please keep your answers eligible.
5. Good Luck!

1 Fibonacci Revisited (20 points)

Last week we showed how to write a simple recursive program for Fibonacci series:

```python
def fib(n):
    return 1 if n <= 1 else fib(n-1) + fib(n-2)
```

But it is exponentially slow due to duplicated computation. Now modify the program so that it would run in linear-time (assume hash operations are constant-time).

```python
def fib(n, fibs=______):
    if not ______________:
        ___________________________________________________________________
        return _________
```

2 Part-of-Speech Tags (40 points)

As we mentioned in last week’s classes, words in natural languages can be tagged by their parts-of-speech, e.g., nouns (NN), determiner (DT), etc. You are now to write a program that will read in a file consisting of word/tag pairs (e.g., “the/DT”), and print for each word (in lexicographical order), the frequencies (in descending order) of each possible tag for that word.

For example, if the input is:

```
the/DT can/NN can/MD can/VB a/DT can/NN
she/PRP saw/VBD the/DT saw/NN
```

your output should be:

```
a DT:1
can NN:2 MD:1 VB:1
saw NN:1 VBD:1
she PRP:1
the DT:2
```
Note that:

1. you can assume all input words are lower-cased, and no word contains "/".
2. if two tags have the same frequency, they should be listed in lexicographical order.
3. Hint: `defaultdict(lambda : <expr>)` returns a dictionary whose default values are `<expr>`. For example, `defaultdict(lambda : [])` is equivalent to `defaultdict(list).

Fill the blanks:

```python
import sys
from collections import defaultdict

words = defaultdict(lambda : ______________________)  # dict of dict

for line in sys.stdin:
    for wordtag in ___________________
        word, tag = wordtag.split("/")
        ____________________________  # increment frequency

for word in sorted(__________________):
    tagfreqs = [_________________________ for tag, freq in __________________________]
    print word, " ".join(["%s:%d" % _____________________
                            for freq, tag in _________________________________])

3 Weird Quicksort (40 points)

We showed in the first class how to rewrite a simple recursive function for quicksort, using Python's list comprehensions. However, I once had a student who came up with the following variant which looks almost identical, but not quite right: she added extra [...] surrounding `qsort(...)`, and as a result, the program is doing something completely different, but surprisingly interesting.

```python
def qsort(a):
    if a == []:
        return []
    left = [x for x in a if x < a[0]]
    right = [x for x in a[1:] if x >= a[0]]
    return [qsort(left)] + [a[0]] + [qsort(right)]
```

Questions:

- (10 pts) Do you see what is this code **really** doing? If you do, jump to question 3 and come back later. Otherwise, after calling `t = qsort ([5, 2, 1, 4, 8, 3])`, what's the result `t`?
2. (5 pts) Visualize this result. You don’t need to include empty lists (\[\]\)).

3. (10 pts) Now do you see what this code is \textit{really} doing? Explain the intuition behind it.

4. (5 pts) What are the usage of empty lists (\[\]\)) in the output? (Hint: they are the potential positions for which operation?)

5. (10 pts) Now that you see the intuition, how to get back the linear-order from t? For example, calling \texttt{sorted(t)} should return the sorted list \([1, 2, 3, 4, 5, 8]\).

Fill in the blanks in the following function.

```python
def sorted(t):
    return [] if t == [] else _______________ + _______ + _______________
```

```python
def sorted(t):
    return [] if t == [] else _______________ + _______ + _______________
```