CSE 399-005 Python Programming, Spring 2006
Homework 2

March 25, 2006

Instructions

1. This assignment is due on Tuesday March 28 at 11:59 PM.
2. You may discuss this HW *in a high level* with another student, but you should write your partner's name in `debrief.txt`. High-level discussions include algorithmic design (but **not** how to implement them), input/output specifications, etc.
3. The other instructions are the same as HW 1.

Problems

In contrast to HW 1, you may submit multiple `.py` files for each problem. But make sure your *main programs* be named as specified.

Problem 1 - Word Frequencies Revisited (15 points)

Main program: `word.py`

In HW 1, Problem 3 asked you to count word frequencies and sort them. Now, with the introdcution of dictionaries, sets, and the built-in `sort` function, life is much easier. So you are to *rewrite* your solution using these new techniques and produce code that is not only more elegant, but also more efficient than your code in HW 1.

The input/output specifications remain unchanged:

```
cat word.in | python word.py > myword.out
```

Problem 2 - Interleaving (20 points)

Main program: `interleave.py`

An **interleaving** of two lists contains all elements from both lists, with the constraint that any two items from the same list are still in the same order.

A **parallel interleaving** is a special case where the items from the two lists alternate in the sequence, i.e., `[ a[0], b[0], a[1], b[1], ... ]`.

Your are to write a program that implements functions `inter(a, b)` and `allinter(a, b)`, where `a` and `b` are two lists, and the former returns the list of parallel interleaving while the latter returns a list containing all interleavings.

Note:

1. the two input lists may or may not be of equal length.
2. you may **not** use any global variable.
3. your code should be no longer than 20 lines.

Example (in the Python interactive shell):

```
>>> from interleave import *
>>> inter([1,2], [3,4,5])
[1, 3, 2, 4, 5]
>>> allinter([1,2], [3,4])
[[1, 2, 3, 4], [1, 3, 2, 4], [1, 3, 4, 2], [3, 1, 2, 4], [3, 1, 4, 2], [3, 4, 1, 2]]
```

For easy debugging, it is recommended that you have the following code in your `interleave.py`:

```
if __name__ == '__main__':
        print inter ([1,2], [3,4,5])
        print allinter ([1,2], [3,4])
```

Now, you can run it in the terminal to test:
`python interleave.py`

It should output the same thing as in the interactive shell. As a larger example, you will get 15 interleavings for
`allinter([1,'a'], ['b', 3, 3, 'a'])`

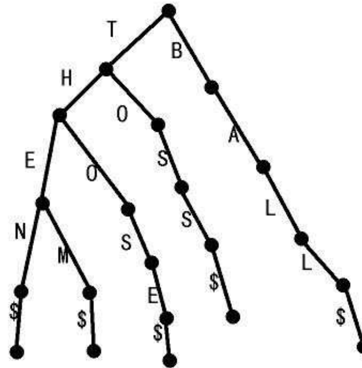
Problem 3 - Trie (55 points)

Main program: `trie.py`

Trie is a special tree structure for efficient manipulation of dictionaries (not in the Python sense). The basic idea is to share the storage of common prefixes. It is similar but faster than binary search trees (BSTs). See the following page for an introdcution.

`http://www.cs.mcgill.ca/~cs251/OldCourses/1997/topic7/`

Here is an example trie, which encodes a dictionary of five words: `then, them, those, toss, ball`. Notice that in trie's internal representation, every word ends with a special symbol `$`. This is to make sure that the trie correctly handles the case when one word is a prefix of another word, e.g., `ball` and `ballet` (assuming `$` never occurs in an English word).



Now you are to use the Python dictionaries to implement a trie. Your program should support the following functions:

1. `search(trie, word)` which returns True if `word` is found in `trie` and False if otherwise.
2. `insert(trie, word)` which inserts `word` into `trie` if it is not already there. (otherwise it does nothing)
3. `remove(trie, word)` which removes `word` from `trie` if it is there. (otherwise it does nothing)
4. `pp(trie)` which *pretty-prints* the trie with indentations. See the example for details.

For testing, you might want to have the following code in your `trie.py`.

```
if __name__ == "__main__":
    trie = {}
    insert(trie, 'ball')
    insert(trie, 'then')
    insert(trie, 'them')
    insert(trie, 'those')
    insert(trie, 'toss')
    insert(trie, 'ballet')
    pp(trie)

    print search(trie, "toss")
    print search(trie, "those")
    print search(trie, "these")
    print search(trie, "ballon")

    remove(trie, 'ballet'); pp(trie)
```

If you run your program in the terminal, it will output the following:

```
root
| b
| | a
| | | l
| | | | l
| | | | | $
| | | | | e
| | | | | | t
| | | | | | | $
| t
| | h
| | | e
| | | | m
| | | | | $
| | | | n
| | | | | $
| | | o
| | | | s
| | | | | e
| | | | | | $
| | o
| | | s
| | | | s
| | | | | $
True
True
False
False
root
| b
| | a
| | | l
| | | | l
| | | | | $
| t
| | h
| | | e
| | | | m
| | | | | $
| | | | n
| | | | | $
| | | o
| | | | s
| | | | | e
| | | | | | $
| | o
| | | s
| | | | s
| | | | | $
```

Note:

1. The pretty-print of a trie starts with a single line of `root`, and each level adds an indentation of `|` .

2. The entries in pretty-print should be in lexicographical order, which implies that the output of `pp(trie)` is *unique*. However, Python dictionaries do not ensure this if you simply do:

   ```
   for letter in trie:
   ```

   So you should use the following to iterate over keys in a sorted order:

   ```
   for letter in sorted(trie):
   ```

3. The special symbol `$` is present in the internal representation and pretty-print, but the `word` argument in functions `insert`, `search`, and `remove` do not have this symbol.

4. The second pretty-print of `trie` corresponds exactly to the example tree in the figure.

Total: 90 (problems) + 10 (styles) = 100 points.

Debriefing

Please answer these questions in `debrief.txt` and submit it along with the programs. There will be 5 points off if you didn't submit this part. If you realized that you forget this part after the deadline, you can just email your `debrief.txt` to TA Bill (`kandylas@cis`).

1. How many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. Are the lectures too fast, too slow, or just in the right pace?
4. Any other comments?
5. If you discussed this HW *in a high-level* with somebody else, who is your partner?