# Exam: Introduction for linguists (LT2102)

Date: October 27, 2010, 8.00 - 13.00

**Course responsible** Markus Forsberg, Språkbanken, Department of Swedish

**Exam accessories** None

**Grade limits** Pass with distinction: 48p, Pass: 28p, Max: 60p

**Note the following:**

- Write readable (unreadable = wrong)!
- Make sure that your code has a clear indentation (if it is unclear, then the least favorable interpretation will be chosen).
- Number the pages, and start every question on a new page.
- Points will be removed for unnecessarily complicated or unstructured solutions.
- I will visit the exam hall around 09.30, so make sure that you have read all questions until then.
- You may use built-in functions and methods, but make sure that you know what they do. If you are unsure, define your own functions.

## Question 1 (6 points)

Define a function `lexical_diversity(words)` that calculates the lexical diversity of a list of words. The lexical diversity is the number of words divided by the number of unique words.

```
>>> lexical_diversity(['same','same','but','different'])
1.3333333333333333
```

## Question 2 (6 points)

1. Define a function `shortest_word_length(words)` that takes a list of words and returns the length of the shortest word.

   ```
   >>> shortest_word_length(['the','sun','shines'])
   3
   ```

2. Define a function `shortest_words(words)` using `shortest_word_length` that returns a list of words containing the words that are of the shortest length.

   ```
   >>> shortest_words(['the','sun','shines'])
   ['the','sun']
   ```

## Question 3 (6 points)

Define a function `is_anagram(word1,word2)` that returns True if `word1` is an anagram of `word2`, False otherwise. `word1` is an anagram of `word2` if it is possible to get `word2` by reordering the letters of `word1`.

```
>>> is_anagram('rise', 'sire')
True
>>> is_anagram('rise', 'sun')
False
```

## Question 4 (6 points)

Define a function `weekdays(weekday)` that returns a list of weekdays, starting with `weekday`.

```
>>> weekdays('Wednesday')
['Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday', 'Monday', 'Tuesday']
>>> weekdays('Saturday')
['Saturday', 'Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

## Question 5 (6 points)

Define a function `censor(words,nasty)` that takes a list of words, and replaces all the words appearing in `nasty` with the word `CENSORED`, and returns the censored list of words.

```
>>> censor(['it','is','raining'], ['raining'])
['it','is','CENSORED']
```

## Question 6 (6 points)

Define a function `syllables(word)` that counts the number of syllables in a word in the following way:

- a maximal sequence of vowels is a syllable;
- a final e in a word is not a syllable (or the vowel sequence it is a part of).

You do not have to deal with any special cases, such as a final e in a one-syllable word (e.g., 'be' or 'bee').

```
>>> syllables('honour')
2
>>> syllables('decode')
2
>>> syllables('oiseau')
2
```

## Question 7 (6 points)

Define a function `count_vowels(text)` that takes a string `text`, counts the vowels in `text` (use a Python dictionary for the counting), and returns the vowel frequency information as a string.

Example:

```
>>> count_vowels('count vowels')
'e: 1\nu: 1\no: 2'
>>> print count_vowels('count vowels')
e: 1
u: 1
o: 2
```

## Question 8 (6 points)

Define a function `utf8_lower(filename)` that reads a utf8-encoded file `filename`, decodes it, and calls `lower()` on the string to turn it into lowercase, then encodes the lowercased string into utf8 again, and write the string to a file named `filename` suffixed with `.lower`.

```
>>> utf8_lower('russian_utf8.txt')
# Creates a file 'russian_utf8.txt.lower' with
# the content of 'russian_utf8.txt' in lowercase.
```

## Question 9 (6 points)

1. Define a class Rectangle according to this example:

```
>>> rec = Rectangle(width=2,length=4)
>>> rec.area()
8
>>> rec.perimeter()
12
```

2. Define a class Square that inherents from Rectangle (you should not redefine `area` and `perimeter` in Square), but is initialized in the following way:

```
>>> sq = Square(side=5)

>>> sq.area()
25
>>> sq.perimeter()
20
```

## Question 10 (6 points)

Consider the following recursive function:

```
def fibonacci(n):
    if n in [0,1]:
        return n
    else:
        return fibonacci(n-1) + fibonacci(n-2)

>>> [fibonacci(n) for n in range(15)]
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377]
```

Redefine this function into a function that does not use recursion.