Exam: Introduction for linguists (LT2102)

Date: December 8, 2010, 13.30 - 18.30

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 14.30, so make sure that you have read all questions until then.
- You may use built-in functions and methods, unless otherwise stated, but make sure that you know what they do. If you are unsure, define your own functions.

Question 1 (6 points)

Define a function `set(lst)` that returns the list `lst` without duplicates. You should not use the built-in `set` to solve this problem.

```
>>> set([1,4,2,5,7,43,2,1,3,4])

[1, 4, 2, 5, 7, 43, 3]
```

Question 2 (6 points)

Define a function `longest_words(words, c)` that gives the list of longest words of `words` starting with letter `c`.

```
>>> longest_words(['a','board','axe','ankh','ache','concord'],'a')

['ankh', 'ache']
```

Question 3 (6 points)

Semordnilap is a word or phrase that spells a different word or phrase backwards. Define a function `semordnilaps(words)` that outputs a list of all words in `words` that are semordnilaps.

```
>>> semordnilaps(['was','a','board','axe','gateman', 'nametag','saw'])
['was', 'gateman', 'nametag', 'saw']
```

Question 4 (6 points)

Define a boolean function `permutation(lst1,lst2)`, that is true if `lst1` is a permutation of `lst2`. A list is a permutation of another if they contain the same elements, but possibly in a different order.

```
>>> permutation([5,2,6,1],[1,2,6,5])
True

>>> permutation([5,2,2,1],[1,2,6,5])
False

>>> permutation([1],[1,1])
False
```

Question 5 (6 points)

Define a function `substitute_with(lst, replacement)`, that substitute all elements in `lst` that occurs as a key in the dictionary `replacement` with the value associated to the key.

```
>>> substitute_with(['a','green','elephant','runs'],
                    {'green':'blue','runs':'walks'})
['a', 'blue', 'elephant', 'walks']
```

Question 6 (6 points)

Define a function `symmetric_difference(lst1, lst2)` that returns the elements occurring in either `lst1` or `lst2`, but not in both.

```
>>> symmetric_difference(['green','blue','red'],['black','red','green'])
['blue', 'black']
```

Question 7 (6 points)

1. Define a class `Student` with the following behavior:

   ```
   >>> student = Student('Smith','MLT',2010)
   >>> student.get_name()
   'Smith'
   >>> student.get_year()
   2010
   >>> student.get_program()
   'MLT'
   ```

2. Define a function `student_report(students)`, where `students` is a list of `Students`, which returns a student information table grendered as a string.

   ```
   >>> student_report([Student('Smith','MLT',2010),
                       Student('Burns','MLT',2010),
                       Student('Flanders','MLT',2010)])
   'Smith 2010 MLT\nBurns 2010 MLT\nFlanders 2010 MLT\n'
   ```

Question 8 (6 points)

Define a procedure `editor(filename,words)` that reads the input word from the user using `raw_input`, and outputs the input word to the file `filename`, one per line, if the word is in `words`. Continue reading and writing until the user inputs `quit`.

```
>>> editor('output.txt',['elephant','cat'])
# user input below
elephant
aardwark
cat
elephant
quit

$ cat output.txt # print content of 'output.txt'
elephant
cat
elephant
```

Question 9 (6 points)

Define a function `pack(lst)` that pack consecutive duplicates of list elements into sublists. If a list contains repeated non-consecutive elements, then they should be placed in separate sublists.

```
>>> pack(['a','a','a','a','b','c','c','a','a','d','e','e','e','e'])
[['a', 'a', 'a', 'a'], ['b'], ['c', 'c'], ['a', 'a'], ['d'],
 ['e', 'e', 'e', 'e']]
```

Question 10 (6 points)

Use `pack` defined in question 9 to implement the so-called run-length encoding data compression method. Consecutive duplicates of elements are encoded as tuples (n,e) where n is the number of duplicates of the element e.

```
>>> encode(['a','a','a','a','b','c','c','a','a','d','e','e','e','e'])
[(4, 'a'), (1, 'b'), (2, 'c'), (2, 'a'), (1, 'd'), (4, 'e')]
```