DALHOUSIE UNIVERSITY *Inspiring Minds*

Winter 2019 (Jan7-Apr8)
Faculty of Computer Science

CSCI 4152/6509 — Natural Language Processing

Assignment 1— Sample Solutions

---

1) (10 marks) Complete the SVN lab tutorial as instructed in the Lab 1 and the lab notes.

For this question, you do not need to submit anything. Your work will be directly checked using the directory '`svnlab`' used for the lab.

Solution:

Follow the lab instructions.

As a result of the lab, it is expected that you created a subdirectory `svnlab` in your SVN repository, which should contain the file `hello.pl`. The file `hello2.pl` should have been added and then deleted.

There should be a record of your changes in the SVN repository as instructed by the lab, including the following log messages:

```
Adding the first version of my sample files.

File hello2 was added by mistake earlier.  It was just
 a backup copy of hello.pl

Use a longer loop iteration

primary

secondary
```

---

2) (22 marks, files in *csuserid*/`lab2`) Complete the Lab 2 as instructed. In particular, you will need to properly:

a) (4 marks) Submit the file 'hello.pl' as instructed.

b) (4 marks) Submit the file 'example2.pl' as instructed.

c) (4 marks) Submit the file 'example5.pl' as instructed.

d) (5 marks) Submit the file 'task1.pl' as instructed.

e) (5 marks) Submit the file 'task2.pl' as instructed.

Notice that the examples from (a) and (b) need to compile; if a syntax error got introduced to an example program by your typing mistake or by introducing incorrect characters through copying and pasting from a pdf file, so that the example program does not compile, it will not be accepted. The lab instructions state that the programs should be tested before being submitted.

Solution:

Follow the lab instructions. The code for submissions a), b), and c) is included in the lab notes.

d) A sample program for `task1.pl`:

```
#!/usr/bin/perl
# Program: task1.pl  Sample solution
use warnings;
use strict;

my $k = 20;
my $str='Use \n for a new line.'."\n";

print $str x $k;
```

e) A sample program for `task2.pl`:

```
#!/usr/bin/perl
# Program: task2.pl  Sample solution
use warnings;
use strict;

sub conc {
    my ($a,$b)=@_;
    if ($a gt $b) {
        return $a.$b;
    } else {
        return $b.$a;
    }
}

print &conc('aaa','ccc');
print "\n";
print &conc('ccc','aaa');
print "\n";
```

---

3) (10 marks) List the levels of NLP and briefly describe all of them in your own words (1–3 sentences for each). Submit your answer as a plain-text file called '`a1q3.txt`'.

Solution:

The levels of NLP discussed in class are, starting from lower-level processing: phonetics, phonology, morphology, syntax, semantics, pragmatics, and discourse.

These are brief descriptions of all of them:

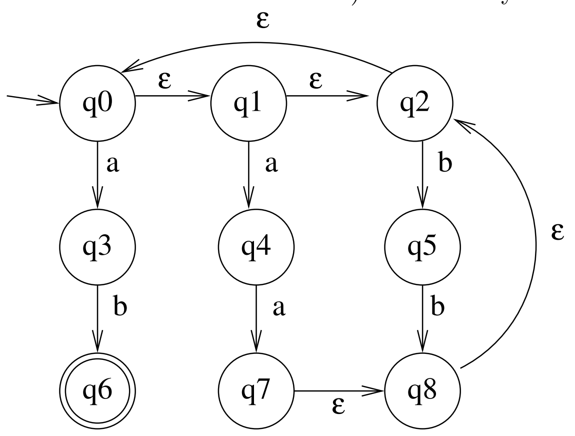
1. **phonetics:** is computer processing concerned with physical sounds of language. It is mostly performed using signal processing methodology. I can be divided into speech generation and speech analysis.
2. **phonology:** is linguistic processing of the sounds of spoken language. It is a higher level than phonetics, mainly concerned with elementary sound units of a language called *phonemes.*

3. **morphology:** is level of processing concerned with the structure of words in a language.
4. **syntax:** is concerned with the sentence structure, i.e., the rules for arranging words within a sentence. One of the main tasks is *parsing,* which is the task of producing a parse tree given a sentence as the input.
5. **semantics:** is interpreting literal meaning of language up to the sentence level.
6. **pragmatics:** is concerned with intended, practical meaning of language.
7. **discourse:** is concerned with language structure beyond sentence level; such as inter-sentence relations, references, and document structure.

---

4) (15 marks) Submit answer to this question in a plain-text file named `a1q4.txt` submited via SVN in the pathname *CSID*/`a1/a1q4.txt`, where *CSID* is your main SVN course directory.

Consider an NFA (Non-deterministic Finite Automaton) described by the following graph:



a) (5 marks) Give three examples of words accepted by this NFA. Briefly describe what language is accepted by this NFA.

Solution:

Three examples of words accepted by this NFA are: `ab`, `aaab`, and `aabbab`.

The language accepted by this NFA is any sequence of strings consisting of pairs of letters ‘aa’ or ‘bb’ and ending with ‘ab’. Just the sequence ‘ab’ is also accepted.

Another way to describe this language is with the regular expression: `/(aa|bb)*ab/`

---

b) (10 marks) Translate this NFA into a DFA using the process discussed in class. Submit your solution as a plain text file named `a1q4.txt` where the DFA is shown as textual table in a format shown below:

```
State      |   a    |    b   |
-----------+--------+--------+
S: q0q1    |  q0q1  |  q0q1  |
-----------+--------+--------+
   q0q1    |  q0q2  |  q0q2  |
-----------+--------+--------+
F: q0q2    |  q0q1  |  q0q1  |
-----------+--------+--------+
```

Exaplanation: Use characters minus, vertical line, and plus to draw the table. The columns correspond to input characters. The DFA states are set of NFA states shows as sequences of states in a sorted order by index (for example, use `q0q1` rather than `q1q0`). use labels `S:` and `F:` to denote start and finish states. If and NFA state is empty set, then use the word '`empty`' to denote it.

Solution:

```
State       |      a       |     b      |
------------+--------------+------------+
S: q0q1q2   |    q3q4      |     q5     |
------------+--------------+------------+
    q3q4    | q0q1q2q7q8   |     q6     |
------------+--------------+------------+
     q5     |    empty     | q0q1q2q8   |
------------+--------------+------------+
 q0q1q2q7q8 |     q3q4     |     q5     |
------------+--------------+------------+
F:   q6     |    empty     |   empty    |
------------+--------------+------------+
    empty   |    empty     |   empty    |
------------+--------------+------------+
  q0q1q2q8  |     q3q4     |     q5     |
------------+--------------+------------+
```

---

5) (10 marks) Submit answer to this question in a file named `a1q5.pl` submited via SVN in the pathname *CSID*`/a1/a1q5.pl`, where *CSID* is your main SVN course directory.

Write a Perl program which does the following:
1) Prints the message: `Enter a positive integer:` with one space after ':' and no newline character.
2) Reads a number from the keyboard. You can assume that the input is always a positive integer.
3) Prints a square pattern of 'X' characters oof width n, where n is an input number. For example, for $n = 5$ it should print:

```
XXXXX
X   X
X   X
X   X
XXXXX
```

The inside of the 'square' is printed with space characters.

Solution:

```
#!/usr/bin/perl
# File: a1q5.pl
# Author: Vlado Keselj
# Date: 6 Jan 2019

print "Enter a positive integer: ";
```

```
my $n = <>;

# It was not required to check if number was not positive, but we will
# play safe and treat 0 and negative numbers as 1.
if ($n <= 1) { print "X\n"; exit; }

my $firstline = 'X' x $n ."\n";
my $middleline = 'X' . ' ' x ($n-2) . 'X' . "\n";
my $middle = $middleline x ($n-2);

print $firstline, $middle, $firstline;

# End of program
# Comment: Rather than using this string-oriented solution, one could
# solve the problem in a style similar to other programming languages
# by using a for-loop for the first and the last line, and a double
# for-loop for the middle part.
```
