T-79.148 **Spring 2004**
Introduction to Theoretical Computer Science
Tutorial 6
Problems

Homework problems:

1. Construct context-free grammars for the following languages:

   (a) $\{a^m b^n \mid 0 \leq m > n\}$

   (b) $\{ucv \mid u, v \in \{a, b\}^* \text{ and } |u| = |v|\}$

   Additionally, give a derivation for the string *aaabb* using your first grammar and another for *abcab* using your second grammar.

2. Let $\Sigma = \{a, b\}$. The *complement* of $w \in \Sigma^*$ is obtained by replacing each a in it with a b and vice versa. For example, $\overline{aba} = bab$. Prove that the language:

   $$L = \{w\overline{w} \mid w \in \Sigma^*\}$$

   is not regular.

3. Construct a context-free grammar that produces all *palindromes* of the alphabet $\Sigma = \{a, b, c\}$, that is, the language:

   $$L = \{w \in \Sigma^* \mid w = w^R\} \ .$$

Demonstration problems:

4. *Pattern expressions* are a generalisation of regular expression used e.g. in some text editing tools of UN*X-type operating systems. In addition to the usual regular expression constructs, a pattern expression may contain string variables, inducing the constraint that any two appearances of the same variable must correspond to the same substring. Thus e.g. aXb^*Xa and $aX(a \cup b)^*YX(a \cup b)^*Ya$ are pattern expressions over the alphabet $\{a, b\}$. The first one of these describes the language $\{awb^n wa \mid w \in \{a, b\}^*, \ n \geq 0\}$. Prove that pattern expressions are a proper generalisation of regular expressions, i.e. that pattern expressions can be used to describe also some nonregular languages.

5. Prove that the language $\{w \in \{a, b\}^* \mid w$ contains equally many a's and b's$\}$ is not regular, and design a context-free grammar generating it.

6. Design a context-free grammar describing the syntax of simple "programs" of the following form: a program consists of nested `for` loops, compound statements enclosed by `begin-end` pairs and elementary operations `a`. Thus, a "program" in this language looks something like this:

```
a;
for 3 times do
begin
   for 5 times do a;
   a; a
end.
```

   For simplicity, you may assume that the loop counters are always integer constants in the range 0,. . . , 9.