Prevodioci i Interpretatori - Januar 2011.
praktični deo

1. Konstruisati DKA za prepoznavanje jezika nad azbukom $\Sigma = \{a, b\}$ čija svaka reč mora da sadrži faktore *aab* i *baa* (obavezno i jedan i drugi!), i zatim napisati C program koji proverava da li uneta niska pripada ovom jeziku.

2. $LL(1)$ gramatikom opisati logičke izraze (sastavljene od iskaznih slova, konjunkcije, disjunkcije i negacije logičkih izraza), a zatim napraviti perl skript koji rekurzivnim spustom proverava da li ulaz zadovoljava gramatiku:

```
p /\ q \/ r
!p \/ ( p /\ q /\ r)
!(p \/ q \/ r)
```

3. Napraviti interpreter za minijaturni skript jezik koji pomalo liči na jezik *PERL*. Jezik odlikuje skalarni tip podataka koji obuhvata cele brojeve i niske karaktera, i nizovni tip koji predstavlja liste skalara. Promenljive u programu se ne deklarišu, ali važi ograničenje da imena skalarnih promenljivih počinju znakom `$` (npr. `$scalar`), dok sve liste pocinju simbolom `@` (npr. `@list`).

   (a) Prvo omogućiti izračunavanje vrednosti izraza nad skalarima. Nad brojevnim skalarima su definisane uobičajene aritmetičke operacije, dok su nad niskama definisane operacije konkatenacije (`.`), množenja brojem (`x`) koje označava konkatenaciju određeni broj puta, kao i operator izdvajanja podniske (`[m,n]`) koji izdvaja `n` karaktera počevši od pozicije `m`. Brojevni skalari se implicitno konvertuju u niske karaktera koje sadrže njihovu tekstualnu reprezentaciju, dok se niske konvertuju u cele brojeve koje opisuju (odnosno 0 ukoliko ne sadrže opis broja).

```
print 23 + 12 - 8 / 4;                          # ispisuje se 33
print "abcd".("efg" x 2)[0,5];                  # ispisuje se "abcdefgef"
print 23 + "abc";                               # ispisuje se 23
print 23 + "12abc";                             # ispisuje se 35
print "abc".12;                                 # ispisuje se "abc12"
```

   (b) Još jedna naredba interpretera je naredba dodele.

```
$broj = 2;                                      # $broj postaje 2
$string = "abc";                                # $string postaje "abc"
@list = ($broj, "def");                         # @list postaje (2, "def")
print(@list);                                   # ispisuje se (2, "def")
$string = ($string x ($broj+1))."ab"[0,1];      # $string postaje "abcabcabca"
print $string;                                  # ispisuje se "abcabca"
```

   (c) Nad listama su definisane funkcija `push` koja ubacuje dati skalar na kraj date liste, `pop` koja vraća poslednji element liste, i skida ga sa steka. Nad listama je definisan i operator indeksnog pristupa `[n]` koji vraća n-ti element liste.

```
push(@list, $string);                           # @list postaje (2, "def", "abcabcabca")
print pop(@list)+13;                            # ispisuje se 13, i @list postaje (2, "def")
print "abc".@list[0];                           # ispisuje se "abc2"
```

Srećno!