



### Prevodioci i interpretatori - 3. test - 7.6.2006.

1. LL(1) gramatikom opisati niz deklaracija funkcija u programskom jeziku C. Npr.

```
int main(int argc, char* argv[]);
char* strdup(char* s);
void f(double a);
```

Za konstruisanu gramatiku odrediti skupove izbora. Napisati PERL skript koji ispisuje sve one linije ulazne datoteke koje predstavljaju ispravnu deklaraciju funkcije. Koristiti tehniku rekurzivnog spusta. [40]

2. Napraviti interpretator za mali jezik iskaznih formula. Formule se grade od iskaznih konstanti T (tačno) i F (netačno) i iskaznih slova [a-z] korišćenjem operatora negacije ~, konjunkcije /\, disjunkcije \/ i implikacije =>.

Ključne reči jezika su **formula** za kojom sledi jedna ili više iskaznih formula, **var** za kojom se navode vrednosti jednog ili više iskaznih slova i **?val** kojom se ispituje vrednost iskazne formule. Vrednost formule se određuje na osnovu vrednosti njenih iskaznih slova. Ukoliko vrednost nekog slova nije definisana, smatra se da je vrednost F. Interpretator sve vreme rada korisniku odgovara ispisivanjem tekstualnih poruka. Npr.

```
>> var p := T, q := F
variable p is set to T
variable q is set to F
>> formula g := (p => q) => (~q => ~p)
formula g is set to ((p)=>(q))=>((~(q))=>(~(p)))
>> ?val g
value of g is T
```

- (a) YACC gramatikom opisati dati jezik. [10]

- (b) Korišćenjem LEX-a kreirati leksički analizator. [8]

- (c) Definisati strukturu podataka u kojoj se beleže vrednosti svih iskaznih slova. YACC program proširiti akcijama koje omogućavaju postavljanje vrednosti iskaznih slova. [7]

```
>> var p := T, q := F
variable p is set to T
variable q is set to F
```

### Prevodioci i interpretatori - 3. test - 7.6.2006.

1. LL(1) gramatikom opisati niz deklaracija funkcija u programskom jeziku C. Npr.

```
int main(int argc, char* argv[]);
char* strdup(char* s);
void f(double a);
```

Za konstruisanu gramatiku odrediti skupove izbora. Napisati PERL skript koji ispisuje sve one linije ulazne datoteke koje predstavljaju ispravnu deklaraciju funkcije. Koristiti tehniku rekurzivnog spusta. [40]

2. Napraviti interpretator za mali jezik iskaznih formula. Formule se grade od iskaznih konstanti T (tačno) i F (netačno) i iskaznih slova [a-z] korišćenjem operatora negacije ~, konjunkcije /\, disjunkcije \/ i implikacije =>.

Ključne reči jezika su **formula** za kojom sledi jedna ili više iskaznih formula, **var** za kojom se navode vrednosti jednog ili više iskaznih slova i **?val** kojom se ispituje vrednost iskazne formule. Vrednost formule se određuje na osnovu vrednosti njenih iskaznih slova. Ukoliko vrednost nekog slova nije definisana, smatra se da je vrednost F. Interpretator sve vreme rada korisniku odgovara ispisivanjem tekstualnih poruka. Npr.

```
>> var p := T, q := F
variable p is set to T
variable q is set to F
>> formula g := (p => q) => (~q => ~p)
formula g is set to ((p)=>(q))=>((~(q))=>(~(p)))
>> ?val g
value of g is T
```

- (a) YACC gramatikom opisati dati jezik. [10]

- (b) Korišćenjem LEX-a kreirati leksički analizator. [8]

- (c) Definisati strukturu podataka u kojoj se beleže vrednosti svih iskaznih slova. YACC program proširiti akcijama koje omogućavaju postavljanje vrednosti iskaznih slova. [7]

```
>> var p := T, q := F
variable p is set to T
variable q is set to F
```

- (d) Definirati klase (strukture) za predstavljanje atomičnih iskaznih formula (iskaznih konstanti i iskaznih slova). Napraviti metode za ispis i određivanje vrednosti ovih iskaznih formula. [10]

```
>> var p := T
variable p is set to T
>> formula f1 := T
formula f1 is set to T
>> ?val f1
value of f1 is T
>> formula f2 := p
formula f2 is set to p
>> ?val f2
value of f2 is T
```

- (e) Definirati klase (strukture) za predstavljanje iskaznih veznika. Napraviti metode za ispis i određivanje vrednosti ovakvih iskaznih formula. [15]

```
>> formula f := (p /\ q) => (p \/ q)
formula f is set to ((p)/\ (q))=>((p)\/(q))
>> ?val f
value of f is T
```

- (f) Omogućiti da već definisane formule učestvuju u izgradnji novih formula (korišćenjem sintakse [name]) [10]

```
>> formula f := p /\ q
formula f is set to (p)/\ (q)
>> formula g := [f] \/ [f]
formula g is set to ((p)/\ (q)) /\ ((p)/\ (q))
```

- (d) Definirati klase (strukture) za predstavljanje atomičnih iskaznih formula (iskaznih konstanti i iskaznih slova). Napraviti metode za ispis i određivanje vrednosti ovih iskaznih formula. [10]

```
>> var p := T
variable p is set to T
>> formula f1 := T
formula f1 is set to T
>> ?val f1
value of f1 is T
>> formula f2 := p
formula f2 is set to p
>> ?val f2
value of f2 is T
```

- (e) Definirati klase (strukture) za predstavljanje iskaznih veznika. Napraviti metode za ispis i određivanje vrednosti ovakvih iskaznih formula. [15]

```
>> formula f := (p /\ q) => (p \/ q)
formula f is set to ((p)/\ (q))=>((p)\/(q))
>> ?val f
value of f is T
```

- (f) Omogućiti da već definisane formule učestvuju u izgradnji novih formula (korišćenjem sintakse [name]) [10]

```
>> formula f := p /\ q
formula f is set to (p)/\ (q)
>> formula g := [f] \/ [f]
formula g is set to ((p)/\ (q)) /\ ((p)/\ (q))
```