Pismeni ispit iz JIP 1.07. 2013

Zadatak 1.

a) Napišite rezultat sljedećeg izraza u Sheme jeziku:

```
>((lambda (x) (* x x x)) 5)
> ..
```

b) Napišite funkciju u jeziku Scheme imena **Kub** koja vraća vrijednost argumenta na potenciju 3, s tim da argument može biti jedan broj ili lista brojeva. (za obraditi listu koristite funkciju **map**)

Zadatak 2. Napišite prijevod u ASMC asemblerski jezik C funkcije Kubiraj(), koja svaki element niza A, od N elemenata, potencira kubno. Primijenite tu funkciju u `main()` funkciji na globalni niz Y. Za lokalne varijable `n` i `x` možete koristite registre procesora.

```
void Kubiraj(int A[], int N)
{
   register n, x;
   for (n = 0; n < N; n++) {
        x = A[n];
       A[n] = x * x * x;
   }
}

int  Y[4] = {7, 21, 22, 7};

int main()
{
     Kubiraj(Y, 4);
     printf("Treci element je: %d", Y[2]);
     return 0;
}
```

Zadatak 3: Napišite leksički analizator u obliku funkcije int GetToken() koja prepoznaje slijedeće tokene i pripadne lekseme:

```
NUM - realni broj koji može biti u običnom i eksponencijalnom formatu
PLUS - '+'
MINUS - '-'
MUL - '*'
DIV - '/'
NL - '\n' (nova linija)
LEFT - '('  (lijeva zagrada)
RIGHT - ')'  (dasna zagrada)
```

Kada funkcija vrati prepoznati token, njegov leksem treba biti spremljen u globalnom stringu char lexeme[32];

Zadatak 4: Napišite rekurzivni parser prema sljedećoj EBNF formi gramatike

```
Naredba : Izraz NL;

Izraz   : Clan  ((PLUS | MINUS) Clan)*;

Clan    : Faktor ((MUL | DIV) Faktor)*;

Faktor  : NUM_TOKEN
        | '(' Izraz ')'
        ;
```

Ova gramatika dozvoljava zapis naredbi oblika: 7 * (7.8+6*7e-3) . Unos naredbe počinje matematičkim izrazom, a završava znakom nove linije (NL token). Nakon toge ispisuje se rezultat izraza.

U izrazima se mogu pojavljivati realni brojevi, operatori (+,-, *, /) i separatori (zagrade, prazna mjesta i tabulatori).

Pretpostavite da vam je na raspolaganju leksički analizator u obliku funkcije int GetToken() , koja vraća token iz ulaznog niza, a njegov leksem sprema u globalnu varijablu char lexeme[]), kao je zadano u prethodnom zadatku.

Funkcije rekurzivnog parsera imaju prototip:

```
void  Naredba();   // ispisuje vrijednost
double Izraz();    // vraća vrijednost izraza
double Clan();     // vraća vrijednost člana
double Faktor();   // vraća vrijednost faktora
```

U svakoj funkciji odredite i semantičke akcije koje rezultiraju vrijednošću koje funkcije vraćaju ili koja se ispisuje u funkciji Naredba().

b) Napišite izraz kojim se dobije lista koja ne sadrži prvi, drugi i treći element prethodne liste L.

c) Koristeći funkciju **filter,** napišite Scheme izraz (ili funkciju) kojim se formira lista od elemenata liste **L** koji su veći od 10.

Zadatak 2. Napišite prijevod u asemblerski jezik sljedećeg C programa:

```
int GetSmaller(int a,int b);
/* funkcija vraća vrijednost manjeg od dva argumenata*/
{
    if(a < b) return a;
    else return b;
}

void main()
{
    int x = 5, y = 4;
    printf("Manji je %d", GetSmaller(x, y));
}
```

Zadatak 3: Napišite specifikaciju leksičkog analizatora za automatsko generiranje funkcije int yylex() pomoću programa LEX, a koja prepoznaje slijedeće tokene i njima pripadne lekseme:

```
NUM - realni broj koji može biti u običnom i eksponencijalnom formatu
PLUS - '+'        (operator zbrajanja)
MINUS - '-'       (operator oduzimanja)
MUL - '*'         (operator množenja)
DIV - '/'         (operator dijeljenja)
EXP - '^'         (operator potenciranja)
NL - '\n'         (nova linija)
LEFT - '('        (lijeva zagrada)
RIGHT - ')'       (desna zagrada)
```

Kostur specifikacije je datoteke "spec.l"

```
{%
enum tokendef {NUM=255, PLUS, MINUS, MUL, DIV, EXP, NL, LEFT, RIGHT};
%}
%%   specifikacija tokena
'\n'  return NL;
......
%%
```

Zadatak 4: Napišite rekurzivni parser prema sljedećoj EBNF formi gramatike

```
Naredba : Izraz NL;
Izraz   : Clan  ((PLUS | MINUS) Clan)*;
Clan    : ExpFaktor ((MUL | DIV ) ExpFaktor)*;
ExpFaktor: Faktor ( EXP Faktor)* ;

Faktor  : NUM_TOKEN
        | '(' Izraz ')'
        ;
```

Ova gramatika dozvoljava zapis naredbi oblika:

7 * (7.8+6*7e-3)^2

Unos naredbe počinje matematičkim izrazom, a završava znakom nove linije (NL token). Nakon toge ispisuje se rezultat izraza. U izrazima se mogu pojavljivati realni brojevi, operatori (+,-, *, /, ^) i separatori (zagrade, prazna mjesta i tabulatori).

Pretpostavite da vam je na raspolaganju leksički analizator, iz prethodnog zadatka, u obliku funkcije int `yylex()` , koja vraća token iz ulaznog niza, a njegov leksem sprema u globalnu varijablu `char yytext[]`), kako je zadano specifikacijom u prethodnom zadatku.

Funkcije i konstante rekurzivnog parsera imaju prototip:

```
enum tokendef {NUM=255, PLUS, MINUS, MUL, DIV, EXP, NL, LEFT, RIGHT};

void  Naredba();     // ispisuje vrijednost
double Izraz();      // vraća vrijednost izraza
double Clan();       // vraća vrijednost člana
double ExpFaktor(); // vraća vrijednost faktora i potencije
double Faktor();     // vraća vrijednost faktora

extern int yylex();      // funkcije za dobavu tokena
extern char yytext[];    // i pripadni string leksema
```

U svakoj funkciji odredite i semantičke akcije koje rezultiraju vrijednošću koje funkcije vraćaju ili koja se ispisuje u funkciji Naredba().

02. 09. 2013.

Zadatak 1:

a) Koji tipovi varijabli se mogu koristiti u jeziku Scheme?

b) Napišite funkciju u jeziku Scheme kojom se iz poznate liste brojeva L1 stvara nova lista L2 koja sadrži članove liste uvećane za 1.

c) Koristeći funkciju **filter,** napišite Scheme izraz (ili funkciju) kojim se formira lista od elemenata liste **L** koji su veći od 10.

Zadatak 2. Napišite prijevod u asemblerski jezik sljedećeg C programa:

```
int GetSmaller(int a,int b, int c);
/* funkcija vraća vrijednost manjeg od tri argumenata*/
{
     if(a<b) {
        if(a<c)  return a;
     }
     else {
        if(b<c)  return b;
     }
     return c;
}

void main()
{
     int x = 5, y = 4, z = 6;
     printf("Manji je %d", GetSmaller(x, y, z));
}
```

Zadatak: Napišite specifikaciju LEX programa za slučaj da funkcija yylex() treba prepo
tokene:

```
FLOAT - realni broj u običnom i eksponencijalnom formatu
INT - integer u decimalnom (123) ili heksadecimalnom for
ID - identifikator varijable - niz znakova (prvi je slo
mogu biti slovo ili broj)
PLUS - '+'          (operator zbrajanja)
MINUS - '-'         (operator oduzimanja)
MUL - '*'           (operator množenja)
DIV - '/'           (operator dijeljenja)
EXP - '^'           (operator potenciranja)
NL - '\n'           (nova linija)
LEFT - '('          (lijeva zagrada)
RIGHT - ')'         (desna zagrada)
PRINT - ?
EQU - '='
```

pecifikacije je datoteke "spec.l"

```
okendef {FLOAT=255, INT, ID, PRINT, EQU, PLUS, MINUS, MUL
, LEFT, RIGHT};

ecifikacija tokena
eturn NL;
.. ovdje dopunite ..............
```

Zadatak 4: Napišite rekurzivni parser (bez semantičkih akcija) prema sljedećoj EBNF definiciji gramatike

```
Naredba : (ID EQU Izraz | PRINT Izraz)  NL;
Izraz   : Clan  ((PLUS | MINUS) Clan)*;
Clan    : Faktor ((MUL | DIV ) Faktor)*;
Faktor   : INT
         | FLOAT
         | ID
         | LEFT Izraz RIGHT
        ;
```

Ova gramatika dozvoljava zapis naredbi oblika:

```
y = x * (7 + y * 7.1e-3)
? y
```

Prvo je naredba pridjele vrijednosti, a drugo je naredba da se ispiše vrijednost izraza iz znaka "?". Unos završava znakom nove linije (NL token). U izrazima se mogu pojavljivati realni i cijeli brojevi, identifikatori, operatori (+,-, *, /, ?) i separatori (zagrade, prazna mjesta i tabulatori).

Pretpostavite da vam je na raspolaganju leksički analizator, iz prethodnog zadatka, u obliku funkcije int `yylex()` , koja vraća token iz ulaznog niza, a njegov leksem sprema u globalnu varijablu `char yytext[]`), kako je zadano specifikacijom u prethodnom zadatku.

Funkcije i konstante rekurzivnog parsera imaju prototip:

```
enum tokendef {FLOAT=255, INT, ID, PRINT, EQU, PLUS, MINUS, MUL, DI
EXP, NL, LEFT, RIGHT};

void  Naredba();     // ispisuje vrijednost
void  Izraz();       // vraća vrijednost izraza
void  Clan();        // vraća vrijednost člana
void  Faktor();      // vraća vrijednost faktora

extern int yylex();       // funkcije za dobavu tokena
extern char yytext[];     // i pripadni string leksema
```