Prevođenje programskih jezika, Januar 1, 19.01.2020.

Napomena: Rešenja čuvajte u direktorijumu formata `ppj.jan1.2020.ime.prezime.indeks.godina` u zasebnim direktorijumima sa nazivima 1 i 2. Na primer, student Marko Marković sa indeksom 27/2013, prvi zadatak čuva u poddirektorijumu 1 direktorijuma `ppj.sep1.2019.Marko.Markovic.27.2013`.
Samo sintaksno ispravan kôd može nositi više od 0 poena. `Makefile` je obavezan deo rešenja.
Preporuka Čuvajte verzije 1. zadatka u zasebnom poddirektorijumu za svaku stavku, na taj način uvek imate verziju koja se prevodila i radila deo zadatka ispravno.

1. Napisati interpetator za varijantu programskog jezika *Omgrofl* koji je neosetljiv na velika i mala slova i podržava sledeću sintaksu:

   (a) Program se sastoji od naredbi koje su pisane svaka u svom redu. Podržane su i prazne naredbe. Program se završava ključnom reči `ktnxbye`.

   (b) Promenljive mogu imati ime oblika `lol`, `lool`, `loool` i slično, ali ne duže od 50 karaktera. Sve promenljive uzimaju celobrojne vrednosti 0-255, bez vodećih nula. Sve moraju biti definisane. U slučaju korišćenja prethodno nedefinisane promenljive program se prekida uz izdavanje obaveštenja korisniku.
   Svaka promenljiva se uvodi definicijom, npr:

   ```
   lol iz 65
   lool Iz LoL
   lol iz looool                    Promenljiva looool nije definisana!
   ```

   Napomena za Lex : Case insensitive mod za deo regexa za koji važi zadaje se sa (?i:ab) za regex [Aa][Bb]. Ukoliko svi treba da su neosetljivi može se koristiti opcija *flex*-a `-i`.

   (c) Realizovati štampanje (naredbom `rofl`) i promenu vrednosti već deklarisanim promenljivama. Podržane operacije su sabiranje, oduzimanje i množenje, pri čemu se mora voditi računa da se ostane u opsegu `[0-255]`. Čak i ako se dobiju negativne vrednosti, moraju se vratiti na opseg. Podržane su i operacije postfiksnog i prefiksnog inkrementiranja i dekrementiranja sa ponašanjem kao u programskom jeziku C/C++ i mogu se primenjivati samo na promenljive. Nije dozvoljeno kombinovanje postfiksnog i prefiksnog operatora nad istom promenljivom. Omogućiti ispis vrednosti nekog izraza, kao i postavljanje promenljive na vrednost nekog izraza. Vrednost se ispisuje kao karakter ASCII tabele.

   ```
   lol iz 65
   lool iz lol+1
   rofl lol                         A
   rofl lool                        B
   lol iz lool*5 +4
   rofl lol                         N
   rofl ++lool                      C
   loool iz 3
   lol iz lool + 2 *loool
   rofl lol                         I
   rofl lol++                       I
   rofl (lol + lool--)*loool-lool   e
   rofl 2 -2*lol                    n
   kTnxBye
   ```

   (d) Naredba za proveravanje ispunjenosti uslova počinje sa ključnom reči `orly?` iza koje sledi uslov i ispisivanje vrednosti izraza. Ispis se uvek zadaje u novom redu. Ukoliko je uslov tačan vrši se ispis vrednosti izraza. Podržani uslovni operatori su:

   - `iz uber` za veće od, `iz liek` za jednako
   - i njihove negacije `iz nope uber` i `iz nope liek`.

   ```
   LOL iz 74
   orly? lol IZ uber 65
   rofl lol++ + 37                  o
   orly? LoL iz nope liek 90
   rofl lol +32                     k
   kTnxBye
   ```

   Napomena: Ovaj ispis ne kodirati preko neterminala za naredbu ispisa već ponovo kodirati u okviru ove naredbe.

   (33 poena)

2. Koristeći **tehniku simulacije rada potisnog automata** implementirati sintaksni analizator koji prepoznaje naredbe definisanja i deklarisanja promenljivih tipa *niska* iz stavke a) 1. zadatka. Promenljivoj se mogu dodeliti vrednosti druge promenljive ili konstantne vrednosti. Program bez naredbi je ispravan program. Nije potrebno vršiti izračunavanja tokom parsiranja ulaza već samo proveriti sintaksu.
   Gramatiku i skupove izbora ostaviti u komentaru. Dozvoljeno je koristiti leksički analizator i `parser.tab.hpp` iz prvog zadatka.

   (12 poena)

Pomoć:

- `std::vector`
  - `size_type size() const;`
  - `iterator begin(); iterator end();`
  - `iterator insert( iterator pos, const T& value );`
  - `iterator erase( iterator pos );`
  - `void push_back( const T& value );`
  - `void pop_back();`
  - `reference operator[]( size_type pos );`
    `const_reference operator[]( size_type pos ) const;`
- `std::map`
  - `std::pair<iterator,bool> insert( const value_type& value );`
    Member type value_type is the type of the elements in the container, defined in map as `pair<const key_type,mapped_type>`
  - `void erase( iterator pos );`
  - `iterator find( const Key& key );`
- `std::string`
  - `string substr (size_t pos = 0, size_t len = npos) const;`
    Returns a newly constructed string object with its value initialized to a copy of a substring of this object.
    The substring is the portion of the object that starts at character position pos and spans len characters (or until the end of the string, whichever comes first).
  - `size_t find (const string& str, size_t pos = 0) const;`
    Searches the string for the first occurrence of the sequence specified by its arguments.
    When pos is specified, the search only includes characters at or after position pos, ignoring any possible occurrences that include characters before pos. The position of the first character of the first match. If no matches were found, the function returns `string::npos`.
- `std::algorithm`
  - `void sort (RandomAccessIterator first, RandomAccessIterator last);`
  - `void reverse (BidirectionalIterator first, BidirectionalIterator last);`

Srećno!