

## Prevođenje programskih jezika

Na Desktopu napraviti direktorijum sa nazivom **PPJ.jun1.ime.prezime.brojIndeksa.godinaUpisa** i u njemu sačuvati svoj rad. Dakle, ako **Pera Perić** sa indeksom **123/2015** polaže ispit potrebno je da na Desktopu napravi direktorijum sa nazivom **PPJ.jun1.Pera.Peric.123.2015** i u njemu da sačuva svoj rad. **Sintaksno neispravni zadaci se ne pregledaju.**  
**Vreme za izradu ispita je 3h. Uslov za polaganje ispita je minimum 22p.**

### Sintaksna analiza naviše

Potrebno je napisati interpreter za zamišljeni programski jezik koji podržava operacije za rad sa grafovima isključivo:

- 1 Interpreter treba da omogući jednostavno definisanje promenljivih nalik na C++ i štampanje njihovih vrednosti na standardni izlaz. Svaki program u ovom programskom jeziku započinje labelom **begin**, nakon čega sledi niz naredbi razdvojenih tačka zarezom. Ključnom rečju “**end**” označava se kraj programa. Pogledati primer u nastavku:

```
begin:  
    graph x;           // definiše prazan graf  
    graph y={1:[1,2], 2:[3,1,4], 3:[4], 4:[]};      // definiše graf sa  
    čvorovima 1,2,3 i 4 predstavljen svojom listom povezanosti  
    graph z=y;  
    print_matrix_d(y); // na cout štampa graf y u formi matrice susedstva  
                      // posmatran kao usmereni graf  
    1 1 0 0  
    1 0 1 1  
    0 0 1 0  
    0 0 0 0  
    print_matrix_u(y); // na cout štampa graf y u formi matrice susedstva  
                      // posmatran kao neusmereni graf  
    1 1 0 0  
    1 1 1 1  
    0 1 0 1  
    0 1 1 0  
end
```

- 2 Interpreter treba da omogući operaciju + kojom se grafovi koji učestvuju u sabiranju spajaju tako što im se za odgovarajuće čvorove uniraju liste suseda. Takođe, potrebno je podržati operaciju množenja dva grafa kojom se dobija presek dva grafa, odnosno graf koji nastaje od zajedničkih čvorova ova dva grafa i veza među njima. Od unarnih operacija potrebno je podržati operaciju ~ kojom se dobija komplementaran graf. Na kraju, potrebno je podržati operaciju grupisanja, to jest, korišćenje zagrade.

```
graph w1 = {1:[4], 4:[10], 10:[1,4]};  
graph w2 = w1 + y; // w2 = {1:[1,2,4], 2:[3,1,4], 3:[4], 4:[10], 10:[1,4]}  
w2 = w1 + {2:[1], 1:[1,2]}; // w2={1:[1,2,4], 2:[1], 4:[10], 10:[1,4]}  
graph w3 = w1*y; // w3 = {1:[1], 4:[]}  
graph w4 = ~w3; // w4 = {1:[4], 4:[1,4]}
```

- 3 Interpreter treba da omogući brisanje čvorova u grafu i dohvatanje liste suseda datog grafa. Kada se čvor obriše, obrišu se i sve grane ka njemu.

```
delete w1 4; // w1 = {4:[10], 10:[1]}  
w2[1]          [1,2,4]
```

4 . Modifikovati interpreter tako da rezultat interpretacije bude reprezentacija programa pomoću sintaksnog stabla. Sintaksno stablo treba da omogući interpretaciju, tj. izvršavanje programa i štampanje stabla na standardni izlaz.

**Napomena:** Makefile je obavazen deo rešenja.

**Napomena 2:** Zadaci 1-4 i zadatak 5 se odvojeno pregledaju. Zbog toga, rešenja zadataka 1-4 i zadatka 5 sačuvajte u odvojenim direktorijumima unutar svog direktorijuma. Na primer, rešenja zadataka 1-4 sačuvajte u direktorijumu „Interaktivni parser“, a rešenje zadatka 5 sačuvajte u direktorijumu „Sintaksno stablo“.