# Programski prevodioci – Projekat 1

Napisati, u jeziku Python, program koji prihvata i izračunava matematičke izraze. Izrazi mogu sadržati proizvoljno mnogo jednocifrenih i višecifrenih brojeva i operacije sabiranja, oduzimanja, množenja i deljenja, kao i operacije poređenja: veće, veće ili jednako, manje, manje ili jednako, jednako i nije jednako. Pod brojevima podrazumevamo samo cele brojeve. Izraz može sadržati i zagrade.

Interpreter podržava tri vrste izraza: infinksni [obavezno], postfiksni [3 poena] i prefiksni [3 poena]. Određeni režim izračunavanja aktivira se unošenjem ključnih reči INFIX, POSTFIX, PREFIX.

Primeri:

~ Plavim slovima je označen text koji se ispisuje na konzoli pre unosa, kao promt, a crnim slovima je označen primer unosa na konzoli.

*INFIX ->* POSTFIX

*POSTFIX ->* 1 2 +

*3*

*POSTFIX ->* INFIX

*INFIX ->* 1 + 2 \* ( 3 – 8 ) / 3

*-2*

*INFIX ->* 3 < 5 > 2

*True*

*INFIX ->* 3 < 5 < 2

*False*

*INFIX ->* EXIT

Jedna sledeća funkcionalnost koju treba podržati je rad sa promenljivama [2 poena]. Promenljiva počinje slovom i može u nastavku sadržati slova, brojeve ili donju crtu. Nakon što se promenljiva prvi put navede bilo gde u izrazu, čuva se u memoriji. U bilo kom sledećem trenutku kada se navede ta promenljiva, automatski se zamenjuje svojom vrednošću, a vrednost promenljivi dodeljujemo znakom za dodelu vrednosti.

Primeri:

*INFIX ->* x

*None*

*INFIX ->* x = 5

*5*

*INFIX ->* x

*5*

*INFIX ->* 1 + x

*6*

*INFIX ->* POSTFIX

*POSTFIX ->* 1 y 2 = + ~ ovo znači 1 + y = 2

*3*

*POSTFIX ->* y

*2*

*POSTFIX ->* y 9 3 + = ~ ovo znači y = 9 + 3

*12*

*POSTFIX ->* y

*12*

Još jedna funkcionalnost koju treba implementirati je funkcija RIM(XXX) [2 poena] koja autmatski pretvara rimske brojeve u arapske i moze se naći bilo gde u izrazu.

Primer:

*INFIX ->* RIM(X)

*10*

*INFIX ->* PREFIX

*PREFIX ->* +RIM(X) 6

*16*

Napomene:

* Prefiksna i postfiksna notacija ne moraju da podržavaju zagrade.
* Ako se u izrazu pojavi barem jedan operator poređenja, rezultat celog izraza je logičkog tipa (True ili False), odnosno, porede se izrazi sa leve i desne strane operatora poređenja. Ako izraz ima više operatora poređenja, rezultat celog izraza je True ako je rezultat poređenja leve i desne strane svakog operatora True. Ako barem jedno poređenje nije tačno, rezultat celog izraza je False. Npr. 5 < 6 < 9 je True zato što je 5 < 6 i 6 < 9, ali izraz 5 < 6 < 2 je False zbog 6 < 2. Ako se u izrazu ne nalazi ni jedan operator poređenja, rezultat izraza je neki broj.
* Zaokruživanje brojeva na izlazu nije obavezno, ali je poželjno jer radimo samo sa celim brojevima.
* U primeru sa promenljivama, znak = je operacija dodele, ali može da se upotrbi i npr. := da bude dodela a = da bude jednako, nije važno koji znak se koristi za koju operaciju, bitno je samo da sve navedene operacije budu podržane.
* Operacija dodele može da se poijavi kao deo zasebnog izraza npr. y = 4 ili da sa svoje desne strane ima izraz npr. y = 1 + 2 \* 8 – 6, gde će vrednost za y biti 11.
* Može da se desi da je operacija dodele deo većeg izraza npr. 5 + y = 2 + 3 \* 6, ovakav slučaj obično nije sintaksno ispravan, ali kod nas desna strana operacije = je do kraja inputa, odnosno izraz 5 + y = 2 + 3 \* 6 se računa kao 5 + ( y = (2 + 3 \* 6) ) gde y “vrati” vrednost 20 pa se dalje računa 5 + 20 , ali ako se izraz dodele nalazi u zagradama npr. 2 + ( y = 3 ) + 1 računalo bi se kao 2 + y + 1, odnosno 2 + 3 + 1 a u memoriji se pamti da y ima vrednost 3, ili npr. 4 \* ( y = 5 + 8 \* 6 ) / 7 bi se racunao kao 4 \* y / 7 a y računamo kao y = ( 5 + 8 \* 6 ) .
* Ako se u inputu navede samo promenljiva, npr. ako je input samo y, to znači da smo upisali u memoriju promenljivu y a jos nismo naveli vrednost za nju. Ako se posle ovakvog slučaja promenljiva primeni negde u izrazu bez dodele, npr. prvo unesemo y, pa unesemo 5 + y, ovakav slučaj može se rešiti tako da y ima neku default vrednost ako se ne navede ništa, ili da je vrednost None što bi bacilo grešku, prihvatiću oba.
* Implementacija infiksnog režima računanja će biti postavljena na materijalima i može se koristiti. Iz tog razloga neću da bodujem infiksni deo, ali on mora da postoji u programu.
* Tokeni mogu biti razdvojeni razmakom ili sa više razmaka, np. 2 + 3 + 4, ali takođe može da se desi da nema razmaka, npr. 1+2+3. Ako ste odvajali tokene po razmacima, ovo može da predstavi problem, tako da računajte da infiskni izraz možda nema razmaka.
* Prefilsni i postfiksni izrazi između brojeva moraju imati razmake npr. 1 3 + je 1 + 3 u infiksnoj i 13+ je (13) + ? što je neispravan izraz i to nećemo da unosimo, ali može da se desi 1 3+, tako da pazite i na ovakve situacije.
* Što se tiče promenljivih, promenljiva mora da počinje slovom i posle ima slova, brojeve, donju crtu ili ništa od toga, tako da RIM može da se tretira kao promenljiva, npr. RIM = 4 je ispravna situacija gde je RIM promenljiva, ali RIM( već ne može da bude, ako se desi RIM( znamo da očekujemo poziv funkcije.
* Negativni brojevi nisam planirala da budu deo zadatka, iako rezultat izraza može da bude negativni broj. Izraze tipa: -3 + 1 ne morate da podržavate, ali ako je neko ipak implementirao, neću da se ljutim :) mada nema dodatnih poena za to.
* Nije dozvoljeno korišćenje rešenja tipa:
  + Funkcija eval()
  + Biblioteke koje pretvaraju rimske brojeve u arapske, ovo je potrebno ručno implementirati.
  + Ugrađene Python biblioteke za Lexer i Parser.

Rok izrade projekta je nedelja pre kolokvijumske. Nije potrebno slanje projekta na mejl već doći na termin koji je po rasporedu za odrećenu grupu i odbraniti projekat na licu mesta.