Drugi kolokvij

Interpretacija programa

29. lipnja 2018.

Ime i prezime: _____

Ovo je "*open book*" kolokvij. Dozvoljeno je korištenje bilo kakvih materijala — bilješke s vježbi/predavanja, vlastiti USBovi s riješenim zadacima, Python help, tutoriali, postovi na online forumima,... — **nastalih prije** kolokvija (npr. dozvoljeno je na *StackOverflow*u naći rješenje nekog zadatka, ali nije dozvoljeno tamo postaviti pitanje kako se rješava neki zadatak). Također, nije dozvoljena komunikacija (razgovor, *chat*, razmjena papira, USBova, ili bilo kakvih materijala) **među** studentima.

U Github repozitoriju kolegija nalazi se zadnja verzija biblioteke `pj.py`. Rješenje zadatka pišite u datoteku `pj_LOOP.py`, te na kraju tu datoteku pošaljite mailom na veky@math.hr, sa Subjectom `IP K2`. Datoteka `pj_LOOP.py` se treba moći izvršiti u Pythonu bez grešaka. Ako imate neki kod koji po Vašem mišljenju pokazuje ideju rješenja, ali iz nekog razloga ne radi, napišite ga u komentar. Korisno je u datoteku uključiti i testove s kojima ste testirali funkcionalnost.

Maksimalno vrijeme rješavanja je 180 minuta. U kolokviju je moguće osvojiti 30 bodova plus 5 bonus bodova.

Veky

RAM-stroj je virtualni procesor (*virtual machine*) s prebrojivo mnogo registara označenih R0, R1, R2, ..., R9, R10, R11,... . U svakom registru može se nalaziti proizvoljni prirodni broj (uključujući 0). Na početku rada RAM-stroja (*reset*) su svi registri inicijalizirani na 0.

LOOP je strojni jezik (*machine code*) za RAM-stroj. LOOP-program je slijed (jedne ili više) instrukcija odvojenih zarezima. Postoje četiri vrste instrukcija (`Rj` označava proizvoljni registar):

- `INC Rj` (inkrement), čije izvršavanje povećava broj u `Rj` za 1
- `DEC Rj` (dekrement), čije izvršavanje smanjuje broj u `Rj` za 1, osim ako je taj broj 0 (tada ne radi ništa).
- `Rj*(program)` (ograničena petlja), čije izvršavanje izvršava `program` onoliko puta koliki je broj bio u registru `Rj` **na početku** izvršavanja petlje (`program` može mijenjati `Rj`, ali to ne mijenja broj izvršavanja petlje).
- `*Rj(program)` (neograničena petlja), čije izvršavanje odgovara uobičajenoj while-petlji (ako je vrijednost `Rj` pozitivna, izvrši se `program` i ponovo testira `Rj`, sve dok vrijednost `Rj` ne postane 0, kada izvršavanje petlje prestaje).

Npr. LOOP-program `R2*DECR2,R3*(INCR2,DECR3)` premješta broj iz R3 u R2.

[5b] Napišite leksički analizator (*tokenizer*) za LOOP-programe. Zanemarite praznine između tokena u ulaznom stringu. Za [-1b] smijete pretpostaviti da su `inc` i `dec` odvojeni od registra prazninom ("`inc R2`"). [5b] Potrebne tipove tokena odredite sami. `inc` i `dec` su *case insensitive*, ali registri se moraju pisati velikim R.

Napišite [5b] beskontekstnu gramatiku i [7b] sintaksni analizator (*parser*) za jezik LOOP. Apstraktna sintaksna stabla odaberite sami. Zagrade su dozvoljene samo oko tijela petlje. Također, zagrade **nisu obavezne** ako tijelo ima samo jednu instrukciju, ali za [-2b] smijete pretpostaviti da jesu.

[2b] Na ASTovima napišite metodu maxreg, koja za dani fragment LOOP-programa vraća maksimalni broj registra koji se u njemu pojavljuje. [6b] Napišite emulator (*runtime environment*) RAM-stroja, u obliku funkcije `run` koja: prima listu prirodnih brojeva [x1,...,xk] i LOOP-program L, resetira RAM-stroj, u `R1` do `Rk` stavi x1 do xk, izvrši L, te kad/ako izvršavanje završi, vrati broj koji se nalazi u `R0`.

[bonus 5b] Napišite LOOP-program *power* koji potencira prirodne brojeve. Dakle, `run([x,y],power)` treba biti x^y, za sve prirodne brojeve x i y. [Za manje bonus bodova, implementirajte zbrajanje odnosno množenje.]