VAJE 8 – Domača naloga

Domača naloga - 1

V sedmi vrstici algoritma za iskanje največjega števila (naloga 8.5) imamo operacijo

Če A_i > *doSedajNajvecji*

Razloži, kaj bi se zgodilo, če bi zgornjo operacijo spremenili v

- Če $A_i \geq$ *doSedajNajvecji* → Dobili bi pravilen rezultat, a za razliko od prej bi algoritem izpisal zadnjo pojavitev največjega števila in njen položaj (če bi bilo pojavitev seveda več).
- Če $A_i <$ *doSedajNajvecji* → Dobili bi napačen rezultat, saj bi nam algoritem namesto največjega izpisal najmanjše število (in položaj) v danem seznamu.

Kaj lahko poveš o pomembnosti uporabe pravilnih primerjalnih operacij (<, >, ≤, ≥, =, ≠) pri pisanju pogojnih in iterativnih operacij?

Odg.: Pri pisanju primerjalnih operacij v pogojnih ali iterativnih stavkih je potrebno biti zelo natančen, saj nam lahko napačna operacija obrne celoten program (primer je lahko kar ta zgoraj, če bi obrnili simbol za večje v manjše, dobimo čisto nasproten rezultat). Prav tako lahko že s pozabljenim ali pa nepotrebnim enačajem pri večje ali enako, dobimo čisto napačen rezultat ali pa (pri iterativnih operacijah) indeksiramo izven področja (»out of range«), zaradi česar se program ne more izvesti.

Domača naloga - 2

Spodaj je predstavljen Evklidov algoritem za iskanje največjega skupnega delitelja dveh pozitivnih celih števil *I* in *J* :

Korak 1. Kot vhod dobiš dve pozitivni celi števili. Večje izmed njiju označi z *I*, manjše pa z *J*.
Korak 2. Deli *I* z *J*, ostanek označi z *R*.
Korak 3. Če *R* ni enak 0, nastavi *I* na vrednost enako *J*, *J* nastavi na vrednost enako *R* in pojdi na korak 2.
Korak 4. Izpiši odgovor, ki je kar enak vrednosti *J*-ja.
Korak 5. Ustavi se.

a) Pojdi skozi algoritem z vhodnima številoma 20 in 32. Po koncu vsakega koraka zabeleži vrednosti *I*, *J* in *R*. Poišči končni odgovor algoritma.

Korak 1.: I = 32, J = 20, R še nima vrednosti → Korak 2.: I = 32, J = 20, R = 12

→ Korak 3.: I = 20, J = 12, R = 12 → Korak 2: I = 20, J = 12, R = 8

→ Korak 3.: I = 12, J = 8, R = 8 → Korak 2.: I = 12, J = 8, R = 4

→ Korak 3.: I = 8, J = 4, R = 4 → Korak 2.: I = 8, J = 4, R = 0

→ Korak 4.: I = 8, J = 4, R = 0 → Korak 5.: I = 8, J = 4, R = 0

Odg.: Končni odgovor je enak končni vrednosti J-ja, torej 4.

b) Ali algoritem deluje pravilno pri vhodu 0 in 32? Opiši, kaj točno se zgodi in spremeni algoritem tako, da vrne primerno sporočilo o napaki.

Odg.: Ne deluje pravilno, saj 32 ne moremo deliti z 0 (nobenega števila ne moramo deliti z 0). Vhodni števili morata biti pozitivni celi števili. Algoritem bi lahko spremenili tako, da bi npr. med korak 1 in korak 2 vrinili Korak 1.2) Če je J >= (večji ali enak) 0, izpiši »Vneseno število ni pozitivno celo število. Prosimo, vnesite veljavni števili« in ponovi korak 1.

Domača naloga - 3

Napiši algoritem za izračun tedenskega honorarnega plačila za delo. Kot vhod algoritem dobi število opravljenih delovnih ur v tednu in urno postavko. Končno plačilo naj se določi tako, da se za vse ure do dopolnjene 40. ure upošteva osnovno (podano) urno postavko, za ure od dopolnjene 40. do dopolnjene 54. količnik 1,50 urne postavke in za ure od dopolnjene 54. dalje količnik 2,00 urne postavke. Algoritem naj izračuna in izpiše višino plačila. Po izpisu plačila naj uporabnika tudi vpraša, ali želi opraviti naslednji izračun. Celoten postopek naj se ponavlja, dokler uporabnik ne odgovori z "NE".

Dobili bi torej podatek o urni postavki, ki ga bom označil z »urna_postavka« in pa število opravljenih delovnih ur, kar bom označil z »stevilo_ur« .

Korak 1: placilo = 0
Korak 2: Dokler stevilo_ur >= (večje ali enako) 40
    Korak 3: Dokler stevilo_ur >= (večje ali enako) 54
        Korak 4: placilo = placilo + (urna_postavka * 2)
        Korak 5: stevilo_ur = stevilo_ur – 1
    Korak 6: placilo = placilo + (urna_postavka * 1,5)
    Korak 7: stevilo_ur = stevilo_ur – 1
Korak 8: placilo = placilo + (urna_postavka * stevilo_ur)
Korak 9: Izpiši placilo
Korak 10: Izpiši » Ali želite opraviti naslednji izračun?«
Korak 11: Če je odgovor »DA«, ponovi korake 1 do 10.
Korak 12: Ustavi se

Domača naloga - 4

Podana je funkcija:

```
(define (mystery input-list)
  (cond ((null? input-list) 0)
    (else ( + 1 (mystery (cdr  input-list))))))
```

a) Kaj bo rezultat ukaza **(mystery (list 3 4 5))** ?

b) Razložite kaj na splošnem dela funkcija.

a) Rezultat bo 3.

b) Funkcija mystery nam pove število elementov vhodnega seznama oz. dolžino seznama. Naredi torej enako kot bi npr. len([3, 4, 5]) v Pythonu.

Funkcija prejme kot argument nek seznam, nato preveri če je ta seznam prazen – če je izpiše oz. doda 0, če ni prišteje 1 in postopek ponovi za seznam brez prvega elementa.

- **Tilen Padovnik, vpisna št.: 63190211**