Zadatak 5.1 - SAT

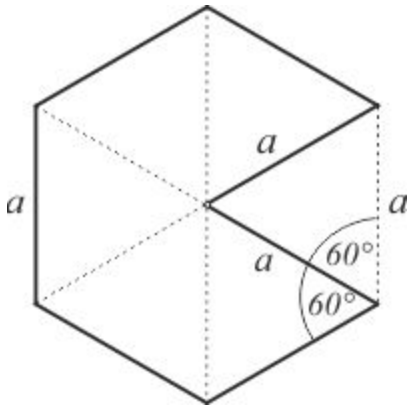
U zadatku se za 60% bodova zahtijevalo poznavanje crtanja jednakostraničnih trokuta te njihovih vanjskih i unutarnjih kutova. Za sve bodove je dodatno bilo potrebno znati nacrtati pravokutnik određenih dimenzija na jednoj stranici trokuta.

Može se primijetiti da orijentacija lika na ekranu nije bitna pa zbog toga poznavanje unutarnjeg kuta trokuta nije bilo nužno za točno riješen zadatak.

Potrebno znanje: osnovne naredbe za kretanje kornjače.

Zadatak 5.2 - PACMAN

U zadatku je za 25% bodova trebalo nacrtati samo tijelo Pacmana. Kao što je u zadatku opisano, tijelo Pacmana sastoji se od dijela šesterokuta kojemu nedostaje najdesnija stranica te još dvije dodatne stranice koje spajaju vrhove nedostajuće stranice sa središtem šesterokuta. Središte se nalazi u sjecištu najduljih dijagonala šesterokuta. Pravilnim crtanjem skice i promatranjem kutova moguće je zaključiti da najdulje dijagonale dijele šesterokut na šest jednakostraničnih trokuta. Iz toga zaključujemo da su duljine stranica koje spajaju vrhove sa središtem šesterokuta dugačke `:a`, kao i stranica šesterokuta, te da su svi kutovi dobiveni crtanjem dijagonala jednaki 60°.



Za potpuno rješenje zadatka treba nacrtati i niz kružnica. Važno je dobro pozicionirati kornjaču za crtanje prve (najljevije) kružnice, na poziciju zadanu u zadatku. Na tu poziciju je najlakše doći pomicanjem iz donjeg desnog vrha šesterokuta za `:a/2` prema gore (točno po stranici šesterokuta koja se ne crta), koristeći naredbu PU da kornjača ne ostavlja trag. Nakon toga u petlji (npr. `REPEAT`) ponavljamo `:n` puta crtanje kružnice i pomak udesno. Kada nacrtamo jednu kružnicu polumjera `:r`, moramo kornjaču pomaknuti za `3*:r` udesno, ne ostavljajući trag, kako bismo se pozicionirali u središte sljedeće kružnice koju je potrebno nacrtati, tako da razmak između kružnica iznosi `:r`.

Potrebno znanje: osnovne naredbe za kretanje kornjače, osnove geometrije (kutovi), petlje.

Zadatak 5.3 - OGRADA

Najprije pokažimo kako se računa pozicija kvadratića na ogradi, odnosno veličina x sa skice iz zadatka. Naime, znamo da kvadratići moraju biti na sredini središnje vertikalne daske, tj. razmak od gornjeg kvadratića do gornjeg ruba daske mora biti jednak razmaku donjeg kvadratića do dna daske. Dakle, prvo je potrebno izračunati kolika je uopće visina središnje daske. Ako imamo, primjerice, 7 dasaka, središnja daska je 4. po redu. To znači da smo 3 puta smanjili visinu daske u odnosu na prvu dasku da bismo dobili visinu 4. daske. Ako ima :n dasaka, a :n je neparan broj, tada je visina središnje daske za `(:n-1)/2*:d` manja od visine prve daske jer je prije središnje daske bilo `(:n-1)/2` dasaka, što znači da visina središnje daske iznosi `:a-(:n-1)/2*:d`. Ako znamo da je svaki od dvaju kvadratića visok `:c` i da je razmak između njih `2*:c`, tada x dobivamo tako da od visine središnje daske oduzmemo `:c+:c+2*:c` i dobiveni iznos podijelimo s 2 (jer samim oduzimanjem dobivamo zbrojene razmak od dna i razmak od vrha). Dio bodova u zadatku moguće je osvojiti i bez provođenja ovog koraka jer je razmak između dasaka jednak nuli.

Jedan od načina za crtanje ograde jest podijeliti crtanje u dvije petlje. Prva petlja se ponavlja `(:n+1)/2` puta crta daske od prve do središnje (uključivo) te u njoj pomoću naredbe MAKE smanjujemo visinu daske za :d u svakom koraku. Druga petlja se ponavlja `(:n-1)/2` puta i crta daske od prve nakon središnje do posljednje, a u njoj povećavamo visinu daske za `:d` u svakom koraku. Dio bodova u zadatku moguće je osvojiti i bez smanjivanja i povećavanja visine daske jer postoje test podaci u kojima je `:d` jednak nuli. Također treba pripaziti da se nakon posljednje vertikalne daske ne crtaju kvadratići sa strane. Sličan se problem javio u zadatku UKRAS na ovogodšnjem školskom natjecanju, pa preporučamo u opisu tog zadatka pogledati kako takav problem riješiti. Moguće je kvadratiće viška obrisati korištenjem naredbe `PE`, no općenito ne preporučamo korištenje te naredbe zbog nepreciznosti.

Potrebno znanje: osnovne naredbe za kretanje kornjače, petlje.

Zadatak 5.4 - DISKO

Promotrimo najprije kako nacrtati jedan prsten koji se sastoji `4*(:n-1)` kvadrata. Ako počinjemo crtati od najljevijeg kvadrata u tom prstenu, trebamo nacrtati niz `:n` kvadrata stranice `:d`, tako da se svaki kvadrat u gornjem desnom kutu dodiruje sa sljedećim u nizu, tj. crtamo kvadrate u smjeru gore-desno. Nakon toga, crtamo niz od `:n` kvadrata u smjeru dolje-desno, tako da se prvi nacrtani kvadrat u novom nizu poklapa sa zadnjim do sada nacrtanim kvadratom. Nakon toga ponovimo postupak za niz kvadrata koji ćemo crtati u smjeru dolje-lijevo i niz kvadrata koji ćemo crtati u smjeru gore-lijevo. Prsten možemo crtati tako da zasebnom petljom nacrtamo svaki od tih nizova kvadrata. Primijetimo da je jedino što se mijenja pri crtanju nizova pomak između crtanja kvadrata. Koristeći tu činjenicu, za crtanje jednog prstena možemo četiri puta ponoviti crtanje jednog niza i pomicanje na odgovarajuću poziciju i usmjerenje za crtanje sljedećeg niza kvadrata. Za implementacijske detalje tog načina rješavanja, proučite službeno rješenje.

Pravilno crtanje odgovarajućeg broja prstena kvadrata donosilo je 40% bodova. Za osvajanje svih bodova, potrebno je bilo još odgovarajuće prstene ispuniti mrežom kvadratića. Ako kvadrate nekog prstena trebamo ispuniti mrežom od :p redaka i :p stupaca (pri čemu je :p veći ili jednak 1), to zapravo znači da trebamo nacrtati `(:p-1)` okomitih i `(:p-1)` vodoravnih crta unutar svakog nacrtanog kvadrata u prstenu. Brojač `:p` na početku postavimo na 0, te nakon svakog nacrtanog prstena, brojač povećamo za `:x`, pazeći da unutar prvog prstena ne crtamo mrežu.

Potrebno znanje: petlje, naredba MAKE, IF.

Zadatak 6.1 - TREŠNJA

Zadatak provjerava osnove programiranja u Logu. Za 40% bodova dovoljno je nacrtati dvije dužine duljine `:d` koje zatvaraju zadani kut. Za osvajanje svih bodova na zadatku bilo je potrebno u produžetku svake dužine nacrtati kružnicu radijusa `:r`. Da bismo to napravili najprije podižemo pero (naredba `PU`), pomičemo se za :r u smjeru dužine, spuštamo pero (naredba `PD`) i konačno crtamo kružnicu radijusa `:r` (naredba `CIRCLE :r`).

Valja naglasiti da u ovom zadatku nije bilo potrebno paziti na rotaciju lika, no prilikom reevaluacije primijetili smo razne pokušaje rotiranja lika tako da mu je os simetrije vertikalna. Svakako savjetujemo da, ako to niste uspjeli napraviti na natjecanju, napravite kod kuće za vježbu, a za inspiraciju se poslužite službenom implementacijom.

Potrebno znanje: osnovne naredbe za kretanje kornjače, crtanje kružnice.

Zadatak 6.2 - ASKLEPIJE

Za 60% bodova u zadatku bilo je potrebno okrenuti se za 180 stupnjeva te `:n/2` puta nacrtati jedan trokut u desnu i jedan trokut u lijevu stranu smanjujući svaki puta stranicu za vrijednost `:b`.

Za ostatak bodova bilo je potrebno imati `FOR` petlju u kojoj se nalazila naredba `IF` koja provjerava je li trokut koji je potrebno nacrtati paran ili neparan po redu. Ako je on paran, potrebno ga je nacrtati ulijevo, a ako je neparan udesno. Na kraju svakog ponavljanja bilo je potrebno stranicu `:a` smanjiti za `:b`.

Potrebno znanje: osnovne naredbe za pomicanje kornjače, provjera parnosti, petlje.

Zadatak 6.3 - ZVIJEZDE

Promotrimo najprije kako nacrtati jednu zvijezdu. Kako se svaka zvijezda sastoji od četiri jednaka dijela, zapravo trebamo četiri puta ponoviti crtanje jedne četvrtine `:m`-terokuta i pravilno pozicioniranje i usmjeravanje prije crtanja sljedeće četvrtine `:m`-terokuta.

Sljedeći isječak koda crta jednu zvijezdu:

```
repeat 4 [repeat :m/4 [fd :d lt 360/:m] fd :d rt 180]
```

Nakon crtanja svake pojedine zvijezde, potrebno se pomaknuti na odgovarajuću poziciju prije crtanja sljedeće zvijezde u retku. Ukoliko piramidu crtamo od najnižeg retka prema višima, nakon što nacrtamo sve zvijezde u pojedinom retku, potrebno je smanjiti brojač koji nam označava koliko zvijezda u retku crtamo (jer svaki redak sazdrži jednu zvijezdu manje od prethodnog) te se pomaknuti na odgovarajuću poziciju za crtanje novog retka, odnosno na vrh prve nacrtane zvijezde iz prethodnog retka. Crtanje redaka ponavljamo dok je broj zvijezda koje je potrebno nacrtati u retku veći ili jednak 1.

Potrebno znanje: ponavljanje uzorka, petlje, naredba MAKE.

Zadatak 6.4 - RIJEČ

Zanemarimo za početak da se ulaznim riječima `:a` i `:b` mogu zamijeniti mjesta. Jedno od mogućih rješenja zadataka jest sastaviti dvije nove riječi, jednu koja se sastoji od `:k` zadnjih znakova riječi `:a`, i drugu koja se sastoji od `:k` početnih znakova riječi `:b`. Potrebno je pronaći najveći broj znakova `:k` za koji se te dvije nove riječi preklapaju, uzevši u obzir da `:k` mora biti manji od broja znakova kraće od dviju riječi `:a` i `:b`. Potrebno je, dakle, napisati pomoćnu funkciju, u službenom rješenju nazvanu `DIO :a :b :r`, koja iz riječi `:r` uzima znakove od pozicije `:a` do pozicije `:b` (uključivo) i vraća novu riječ koja se sastoji od tih znakova. U ovoj funkciji potrebno je koristiti naredbu `WORD`. Sada možemo napisati funkciju `RIJESI :a :b` koja pronalazi točno rješenje ako je prva riječ `:a`, a druga riječ `:b`. U funkciji imamo petlju koja za svaki mogući broj znakova `:k` uspoređuje preklapa li se dio s kraja prve riječi s dijelom s početka druge riječi. Ako nađemo takav `:k`, vraćamo riječ koja se sastoji od cijele prve riječi i ostatka druge riječi koji ne ulazi u preklapanje. Ako takav `:k` ne nađemo, vraćamo samo spojene riječi `:a` i `:b`.

Prisjetimo se da riječima `:a` i `:b` moguće i zamijeniti mjesta. U konačnoj funkciji `RIJEC :a :b` funkciju `RIJESI` možemo pozvati dva puta i zapamtiti oba rješenja, jednom `RIJESI :a :b`, a drugi put `RIJESI :b :a`. Ako su dobivena dva rješenja različite duljine (provjera naredbom `COUNT`), vraćamo ono kraće. Ako su istih duljina, moramo vratiti ono rješenje koje je prvo po abecedi, što možemo provjeriti naredbom `BEFORE?`.

Potrebno znanje: petlje, riječi.

Zadatak 7.1 - SEDAM

Za osvajanje 20% bodova na ovom zadatku, dovoljno je bilo nacrtati sedmerokut stranice `:a`.
Za osvajanje 40% bodova na ovom zadatku, dovoljno je bilo nacrtati sedmerokut stranice `:a` kojemu se na polovištima stranica nalaze crte duljine `:b`.

Za osvajanje svih bodova, potrebno je bilo na vrhu crte duljine `:b` nacrtati još i simetričan niz od 7 kružnica. Jedan od mogućih načina crtanja tog niza je da najprije nacrtamo središnju kružnicu polumjera `:c1`, pomaknemo se udesno za `:c1+:c2` te nacrtamo kružnicu polumjera `:c2`, zatim se pomaknemo udesno za `:c2+:c3` te nacrtamo kružnicu polumjera `:c3`, te se pomaknemo udesno za `:c3+:c4` i nacrtamo kružnicu polumjera `:c4`. Nakon toga se vratimo u središte prve nacrtane kružnice te ponovimo postupak crtajući ulijevo. Nakon što nacrtamo svih 7 kružnica, vratimo se u polovište stranice, nacrtamo preostalu polovicu stranice i okrenemo se za kut `360/7` te ponovimo isti postupak.

Potrebno znanje: osnovne naredbe za pomicanje kornjače, crtanje kružnice, ponavljanje uzorka.

Zadatak 7.2 - ZGRADE

Za 40% bodova u zadatku bilo je moguće naredbom `IF` pokriti slučajeve za `:n` manji od 5.

Zadatak rješavamo ukrštenim petljama. Vanjska petlja pratit će broj katova koji bi trebala imati svaka zgrada i pomicati kornjaču na položaj iduće zgrade, dok će unutarnja crtati svaku pojedinu zgradu. Tako će unutarnja petlja prvi put crtati zgradi s jednim katom, drugi puta s dva kata, itd. sve dok razlika između ukupnog broja katova i varijable koja pamti broj nacrtanih katova ne bude manja ili jednaka predviđenom broju katova zgrade koja se iduća crta. Kada to bude slučaj, potrebno je nacrtati broj katova jednak toj razlici i završiti izvođenje programa.

Potrebno znanje: osnovne naredbe za pomicanje kornjače, petlje, naredba IF

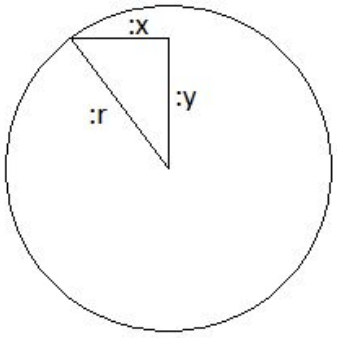
Zadatak 7.3 - ZIDINE

Za 20% bodova je u zadatku bilo dovoljno nacrtati pravilni n-terokut zadane duljine stranice. S obzirom na jednostavnost ovog podzadatka, uvijek se preporuča pažljivo čitanje sekcije o bodovanju.

Za 60% bodova se tražilo crtanje kružnica u vrhovima n-terokuta. Najjednostavnije rješenje je izračunati skraćenu duljinu zidova `(:d-2*:r)` te se pomoću `PU` i `PD` pozicionirati kako nam zidovi ne bi ulazili u kule.

Prva dva podzadatka su bila uvod u konačni podzadatak koji je nosio sve bodove. Za sve bodove je potrebno zidove proširiti za `:x` na svaku stranu. Postoji nekoliko načina na koje se ovo moglo riješiti, mi ćemo spomenuti dva načina.

1) Iz Pitagorina poučka možemo poznavajući `:x` i `:r` izračunati `:y`, tj. za koliko se moramo pomaknuti kako bi se točno pozicionirali na rub kule.



$$y = \sqrt{r^2 - x^2}$$

   Nakon što smo izračunali `:y`, preostaje nam povući crtu za zid. Službeno rješenje za to koristi koordinatnu grafiku da se zapamte potrebne pozicije, ali se isto tako moglo direktno izračunati duljinu zida.
2) Koristeći `FILL` i mijenjanjem boja se može u potpunosti izbjeći ikakvo računanje. Zidove prvo nacrtamo predugačko crnom bojom tako da bi nam ulazili u kule, potom kule nacrtamo nekom drugačijom bojom (npr. crvenom) te onda koristeći `FILL` ispunimo kulu crnom bojom. Nakon toga ponovno koristeći `FILL` ispunimo sve bijelom bojom. To će nam skratiti zidove za točno onoliko koliko je potrebno da ne ulaze u kule. Potom ponovno nacrtamo kulu crnom bojom kako bi slika bila točna.

Potrebno znanje: Pitagorin poučak, osnove koordinatne grafike, naprednije bojenje.

Zadatak 7.4 - CIJENA

Da bismo izračunali cijenu neke liste, moramo znati kako ona izgleda i na kojoj se dubini nalazi u originalnoj listi. Napišimo, stoga, funkciju `rek :l :dub` koja računa cijenu liste `:l` koja se nalazi na dubini `:dub`.

Iz teksta zadatka znamo da ta cijena odgovara sumi cijena njenih elemenata pomnoženih sa dubinom na kojoj se nalazi, s time da cijena nekog broja odgovara tom broju, a cijena neke liste se ponovno pokorava ovoj definiciji, a njena dubina je za jedan veća od dubine trenutne liste. Odnosno, ako lista `:l` sadrži neku listu `:k` kao podlistu, a funkcija `rek :l :dub` računa cijenu liste `:l` koja se nalazi na dubini `:dub`, tada će poziv funkcije `rek :k (:dub+1)` izračunati cijenu podliste `:k`. Koristeći ovu rekurzivnu relaciju, implementacija funkcije `rek :l :dub` postaje jednostavna i prirodno se nameće.

```
to rek :l :dub
  localmake "ret 0
  foreach :l [
    ifelse numberp ? [
      make "ret :ret+?
    ] [
      make "ret :ret+(rek ? :dub+1)
    ]
  ]
  op :ret*:dub
end
```

Konačno, ukupna cijena naše početne liste odgovara pozivu ove funkcije s parametrima `:l` i 1 pa je sama implementacjia jednostavna i izgleda ovako:

```
to cijena :l
  op rek :l 1
end
```

Potrebno znanje: osnovne naredbe za rad s listama, rekurzija.

Zadatak 8.1 - BATERIJA

Zadatak se mogao riješiti za 60% bodova bez bojenja baterije. Bilo je potrebno nacrtati veliki pravokutnik, pomaknuti se na trećinu gornje stranice i nacrtati mali pravokutnik te nacrtati vodoravnu crtu koja spaja bočne stranice većeg pravokutnika na visini `:p*:a/100`.

Za dodatnih 40% bodova bilo je potrebno i pravilno obojiti prostor ispod vodoravne crte. Naredbom IFELSE bilo je potrebno provjeriti je li `:p` veći od 15 i ovisno o tome postaviti boju ispune na “GREEN ili “RED. Potom je trebalo pomaknuti kornjaču unutar prostora ispod vodoravne crte s podignutim perom pazeći da ne izađemo van iz baterije ili prostora koji je potrebno obojiti - u zadatku je bilo navedeno da će prostor biti visine najmanje 5 piksela pa se kornjača mogla pomaknuti za 1, 2, 3 ili 4 piksela iznad dna ili ispod vodoravne crte kako bi bili sigurni da se nalazimo u prostoru kojeg je potrebno obojiti. Prostor bojimo naredbom `FILL`.

Bilo je potrebno paziti i na slučaj kada je :p=0. Tada, naime, ne postoji prostor koji je potrebno obojiti pa je dovoljno nacrtati samo obris baterije.

Potrebno znanje: osnovne naredbe za pomicanje kornjače, naredba `IF`, naredba `FILL`.

Zadatak 8.2 - ARHIMED

Zadatak možemo riješiti koristeći dvije petlje. Vanjska petlja se ponavlja `:n+1` puta jer toliko nizova kružnica crtamo, ako i prvu kružnicu smatramo nizom od jedne kružnice. Unutarnja petlja se ponavlja :m puta, gdje je :m broj kružnica u nizu koje crtamo jednu do druge. U prvom koraku je :m jednak 1, u drugom 2, u trećem 4, i tako dalje. Vidimo da u vanjskoj petlji moramo broj :m nakon svake iteracije množiti s 2 kako bismo dobili točan broj kružnica u nizu u sljedećem koraku jer se unutar svake kružnice u sljedećem koraku crtaju dvije manje. Ovaj način rješavanja implementiran je u službenom rješenju.

Za vježbu, zadatak je moguće riješiti koristeći rekurziju, što je možda način rješavanja koji se prirodno nameće kada vidimo željeni izlaz, ali ne i najjednostavniji. Vidimo da se ponavlja uzorak crtanja dviju kružnica jedne pokraj druge, u različitim veličinama. Jedno od mogućih rješenja rekurzijom jest sljedeće:

```
to kruznice :n :d
    circle :d
    if :n=0 [stop]
    pu lt 90 fd :d/2 rt 90 pd
    kruznice :n-1 :d/2
    pu rt 90 fd :d lt 90 pd
    kruznice :n-1 :d/2
```

```
    pu lt 90 fd :d/2 rt 90 pd
end
```

Potrebno znanje: osnovne naredbe za kretanje kornjače, petlje.

Zadatak 8.3 - ZMAJ

Dio bodova na ovom zadatku bio je ostvariv tako da ručno generiramo riječi koje opisuju određene zmajolike krivulje i te slučajeve pokrijemo (hardkodiramo) u kodu. Očekivano rješenje zadatka sastoji se od dva dijela - izgradnje riječi koja odgovara zmajolikoj krivulji n-tog stupnja i crtanja zmajolike krivulje na temelju zadane riječi.

Oba dijela zadatka mogu se riješiti iterativno (bez korištenja rekurzija), no kako bismo koncept rekurzije što više približili natjecateljima, odlučili smo u ovom opisu i službenoj implementaciji demonstrirati rekurzivan pristup rješavanju ovog problema.

Krenimo najprije od naše glavne procedure koja crta zmajoliku krivulju na temelju riječi koja je generira. Implementacija ove procedure izgleda ovako:

```
to zmaj :n :d
  crtaj (izgradi :n) :d
end
```

Ostaje nam implementirati proceduru `crtaj :w :d` koja crta zmajoliku krivulju opisanu pomoću riječi `:w`, a duljina jedne dužine iznosi `:d`, te funkciju izgradi `:n` koja gradi riječ koja odgovara zmajolikoj krivulji stupnja n-tog stupnja.

Implementirajmo funkciju `crtaj :w :d`. Najprije se trebamo pomaknuti unaprijed za `:d`, a zatim se okrenuti za 90 stupnjeva u smjeru koje nam govori prvo slovo riječi `:w`. Nakon što to napravimo, potrebno je isti postupak ponoviti za sva ostala slova riječi `:w`. To možemo napraviti tako da ponovno pozovemo istu funkciju s parametrima `(bf :w)` i `:d`. Naravno, potrebno je tada dodati i zaustavni uvjet kada je riječ prazna kako ne bismo završili u beskonačnoj rekurziji. Implementacija je elegantna i izgleda ovako:

```
to crtaj :w :d
  fd :d
  if empty? :w [stop]
  ifelse (first :w)="D [rt 90] [lt 90]
  crtaj bf :w :d
end
```

Ostaje nam još samo implementirati funkciju izgradi `:n`. Definicija funkcije slijedi iz samog teksta, naime, riječ koja odgovara zmajolikoj krivulji n-tog stupnja odgovara riječi krivulje stupnja `(:n-1)` kojoj na kraj dodajemo slovo D i obrnuti komplement te riječi. Komplement neke riječi je riječ kojoj su sva slova D zamijenjena slovima L i obrnuto. Kako rekurzija ne bi išla u beskonačnost, potrebno je dodati zaustavni uvjet. Stoga, definicija funkcije `izradi :n` jest:

```
to izgradi :n
  if :n=0 [op "]
  op (word izgradi :n-1 "D reverse komplement izgradi :n-1)
end
```

Kao što vidite, koristimo i pomoćnu funkciju komplement :w koja nam vraća komplement riječi :w. Implementaciju ove funkcije ostavljamo čitatelju za vježbu, a njenu rekurzivnu implementaciju možete pogledati u službenom rješenju.

Potrebno znanje: rad s riječima, podjela problema na potprobleme, petlje ili rekurzija

Zadatak 8.4 - SPLIT

Za 10% bodova se može u potpunosti zanemariti koja je riječ na ulazu te samo generirati izlaz na temelju liste brojeva. Dovoljno je poznavanje neke petlje i naredbe `WORD` koja spaja dvije riječi u jednu koju u našem slučaju koristimo za spajanje potrebnog broja slova "`a`" zajedno.

```
to split :r :l
  make "sol []
  for [i 1 [count :l] 1] [
    make "sol lput (uzmi (item :i :l)) :sol
  ]
  op :sol
end

to uzmi :x
  make "rez "
  repeat :x [make "rez word :rez "a]
  op :rez
end
```

Ovaj osnovni kod možemo proširiti da dobijemo 40% bodova tako da umjesto slova "`a`" pamtimo koje slovo sljedeće moramo uzeti iz zadane riječi te provodimo isti algoritam.

Za sve bodove je potrebno poznavanje funkcioniranje rekurzija te lokalnih i globalnih varijabli u Logu. Prvo ćemo iz glavnog programa izdvojiti novu funkciju `rek :l` kako bismo u glavnom programu mogli inicijalizirati potrebne varijable:

```
to rek :l
  localmake "sol []
  for [i 1 [count :l] 1] [
    ifelse listp (item :i :l) [
      make "sol lput (rek (item :i :l)) :sol
    ][
      make "sol lput (uzmi (item :i :l)) :sol
    ]
  ]
  op :sol
end
```

Ovaj kod čini bazu našeg programa i rekurzije. Prvo inicijaliziramo varijablu `:sol` kao praznu listu, ali pazeći da bude lokalna samo trenutnoj razini rekurzije. Nakon toga za svaki element naše liste provjeravamo je li lista ili broj. U slučaju kada je lista, mora se ponovno rekurzivno pozvati funkcija `rek`. U slučaju kada je broj, normalno uzimamo potreban broj slova iz naše riječi.

Pogledajmo sada kako smo promijenili glavnu funkcije `split` i `uzmi`:

```
to split :r :l
  make "tr 1
  make "rijec :r
  op rek :l
end

to uzmi :x
  make "rez "
  repeat :x [
    make "rez word :rez (item :tr :rijec)
    make "tr :tr + 1
  ]
  op :rez
end
```

Možemo vidjeti da u funkciji `split` inicijaliziramo varijable `:tr` i `:rijec` na neke početne vrijednosti. Koristeći normalnu `MAKE` naredbu (u odnosu na `LOCALMAKE`) smo te dvije varijable učinili globalnim i mogu se koristiti i u drugim funkcijama, tj. u našem slučaju funkciji `uzmi`. Funkcija uzmi se nije drastično promijenila i u suštini radi isto što i za drugi podzadatak. Ovime smo u potpunosti riješili zadatak. Konačni kod nije previše dugačak, ali zahtijeva dobro razumijevanje onoga što se u njemu događa.

Potrebno znanje: liste, rekurzije, funkcije