# Zadatak 5.1 - Puzzle

Primijetimo da se puzzle sastoje od ravnih rubova, te rubova koji na sredini sadrže udubinu ili izbočinu u obliku kvadrata. Ako je potrebno nacrtati rub koji sadrži udubinu, onda je potrebno 5 puta pomaknuti kornjaču unaprijed za `:a`, a nakon svakog pomaka se okrenuti za 90°, i to redom prvo udesno, pa ulijevo, pa ponovno udesno te još jednom ulijevo. Ako je potrebno nacrtati rub koji sadrži izbočinu, onda se također pomičemo 5 puta unaprijed za `:a` i okrećemo za 90°, ali je redoslijed okreta drugačiji. Prvo se okrećemo ulijevo, pa dva puta udesno te naposljetku još jednom ulijevo. Za crtanje cijele slagalice potrebno je nacrtati ravne rubove duljine `3*:a` na dnu svake puzzle i na lijevom i desnom rubu slagalice, te nacrtati rubove s udubinama i izbočinama na odgovarajućim puzzlama.

Za implementacijske detalje pogledajte službeno rješenje.

# Zadatak 5.2 - Cvijet

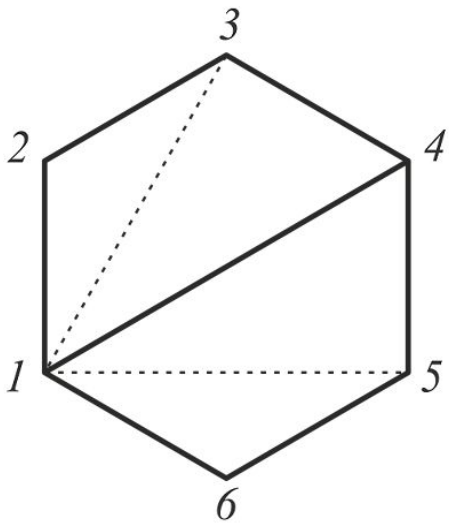
Promotrimo najprije kako bismo riješili jednostavniju verziju ovog zadatka u kojoj crtaju sve stranice mnogokuta, uključujući one koje dijele `:m`-terokuti i `:n`-terokut. Tada bismo `:n` puta ponovili crtanje `:m`-terokuta ulijevo te bismo se pomaknuli naprijed za `:d` i okrenuli za `360/:n` udesno.

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

Stranica koju dijele `:m`-terokuti i `:n`-terokut je prva stranica koju nacrtamo prilikom crtanja `:m`-terokuta. S obzirom na to da ne želimo da ona bude vidljiva, možemo nju nacrtati bez ostavljanja traga, a preostalih `:m`-1 stranica `:m`-terokuta nakon toga nacrtati kao i inače.
Pritom pazimo da ne ostavljamo trag kada se pomičemo na poziciju iz koje počinjemo crtati sljedeći `:m`-terokut.

```
repeat :n[
  pu fd :d lt 360/:m pd
  repeat :m-1[fd :d lt 360/:m]
  pu fd :d rt 360/:n pd
]
```

# Zadatak 5.3 - Dijagonale




Promotrimo najprije kako pronaći najdulju dijagonalu (ili dijagonale) koja izlazi iz nekog vrha. S obzirom na to da dijagonala spaja dva nesusjedna vrha u mnogokutu, jasno je da će najdulja dijagonala biti ona koja spaja vrh s njemu najudaljenijim vrhom. Pritom nam je intuitivno jasno da će najudaljeniji vrh biti onaj do kojeg je potrebno proći kroz najviše vrhova, krećući se u onom smjeru u kojem nam je traženi vrh bliži. Primjerice, za šesterokut na skici vrijedi da na putu od vrha 1 do vrha 4 moramo proći kroz tri vrha, a na putu od vrha 1 do vrha 5 moramo proći kroz samo dva vrha.

Kod mnogokuta s parnim brojem vrhova, taj najveći broj vrhova kroz koje moramo proći će biti jednak `:n/2`. Kod mnogokuta s neparnim brojem vrhova, najudaljeniji vrh neće biti jedinstven. Najudaljeniji vrhovi vrhu A će biti oni koji se nalaze udaljeni `(:n-1)/2` vrhova, gledajući u smjeru kazaljke na satu i smjeru obrnutom od kazaljke na satu, odnosno vrhovi koji su udaljeni `(:n-1)/2` i `(:n+1)/2` u smjeru kazaljke na satu od vrha A.

Promotrimo sada kako najjednostavnije nacrtati dijagonale. Rješavanje zadatka možemo podijeliti na slučaj kad je `:n` paran i kad je `:n` neparan. Ako je `:n` neparan, potrebno je nacrtati one dijagonale koje izlaze iz prvih `:n/2` vrhova, jer one ulaze u preostalih `:n/2` vrhova pa ćemo time nacrtati sve tražene dijagonale. Stoga `:n/2` puta ponavljamo sljedeće: pozicioniramo se u vrh mnogokuta, te spremimo poziciju vrha u varijablu `:x`. Zatim se `:n/2` puta pomičemo po susjednim vrhovima u smjeru kazaljke na satu, te se nakon toga korištenjem naredbe `SETPOS` pomaknemo na poziciju `:x`, pazeći da pritom ostavimo trag. Time ćemo nacrtati jednu dijagonalu. Zatim se pomaknemo u sljedeći susjedni vrh i ponovimo isti postupak.

```
repeat :n/2[
    make "x pos make "h heading
    repeat :n/2[fd :d rt 360/:n]
    setpos :x seth :h
    fd :d rt 360/:n
]
```

Korištenjem naredbe `SETH` osiguravamo da smo se pravilno usmjerili pri povratku na trenutni vrh. Isti efekt bismo postigli i da smo se nakon povratka na poziciju `:x` okrenuli ulijevo `:n/2` puta za `360/:n`, jer smo se toliko puta okrenuli udesno pri pomaku do najudaljenijeg vrha.

Kod mnogokuta s neparnim brojem vrhova moramo izvršiti dva spajanja trenutnog vrha s preostalima, odnosno nacrtati dvije dijagonale iz trenutnog vrha. U svakom koraku spremimo poziciju trenutnog vrha u varijablu `:x`. Zatim se `(:n-1)/2` puta pomaknemo u susjedni vrh, te spremimo poziciju vrha u kojem smo završili u varijablu `:y`, te se još jednom pomaknemo u vrh susjedan tom vrhu. Zatim se pozicioniramo na poziciju `:x`, pa na poziciju `:y`. Time smo nacrtali obje dijagonale koje izlaze iz vrha kojeg trenutno obrađujemo, a onda je potrebno još se jednom pozicionirati na poziciju `:x` kako bismo mogli obraditi sljedeći vrh mnogokuta.

U natjecateljskom programiranju često nailazimo na probleme koje valja rastaviti na slučajeve koje zasebno rješavamo. Također, nerijetko se dogodi da te slučajeve možemo objediniti koristeći neku dosjetku ili implementacijski trik. U ovom konkretnom slučaju, minimalnim izmjenama procedure koja rješava zadatak za parni `:n`, pokrit ćemo i neparan slučaj te tako u potpunosti riješiti zadatak:

```
repeat :n [
  make "x pos make "h heading
  repeat (int :n/2) [fd :d rt 360/:n]
  setpos :x seth :h
  fd :d rt 360/:n
]
```

# Zadatak 5.4 - Domino

Promotrimo najprije kako najjednostavnije nacrtati jednu domino pločicu. Postoji šest različitih rasporeda kružnica unutar svake polovice domino pločice. Stoga je korisno napraviti proceduru koja će crtati jednu polovicu domino pločice za zadani broj kružnica koji se u njoj treba nalaziti.

Ukoliko s `:x` označimo broj kružnica koje je potrebno nacrtati unutar polovice pločice, onda možemo korištenjem `IF` naredbe usporediti `:x` s brojevima od 1 do 6 te za svaki mogući broj nacrtati traženi raspored kružnica. Elegantnije bi bilo da, umjesto da pišemo zaseban dio koda za svaki mogući broj kružnica, kornjačom prođemo sve pozicije središta kružnica bez ostavljanja traga. Svaki put kad dođemo u neko središte kružnice provjerimo trebamo li ovdje nacrtati kružnicu. Primjerice, kružnicu u samom središtu kvadrata trebamo nacrtati ako je `:x` jednak 1, 3 ili 5. Kada dođemo na poziciju središta te kružnice, usporedit ćemo `:x` s 1, 3 i 5. Ako je `:x` jednak nekom od tih brojeva, nacrtat ćemo kružnicu. Za implementacijske detalje ovog pristupa proučite službeno rješenje.

Sada lako možemo nacrtati jednu domino pločicu. Ako s `:a` označimo broj kružnica u lijevoj polovici, a s `:b` označimo broj kružnica u desnoj polovici, dovoljno je dva puta pozvati proceduru koja crta jednu polovicu, redom s argumentima `:a` i `:b`. Na početku je `:a` jednak 1, a `:b` jednak 2. Nakon crtanja svake pločice povećamo obje varijable za 1. Kad neka varijabla nakon povećavanja postane jednaka 7, postavimo je ponovno na 1. Ovime smo osigurali da

nakon pločice s oznakom 6 ponovno crtamo pločicu s oznakom 1. Crtanje pločica ponavljamo ukupno `:n` puta, pazeći pritom da se pomaknemo na odgovarajuće mjesto prije početka crtanja sljedeće pločice.

# Zadatak 6.1 - Ekran

U ovom zadatku bilo je potrebno nacrtati manji ekran (postupak se mogao skratiti korištenjem naredbe `REPEAT` za crtanje pravokutnika), dignuti pero i pomaknuti se za `:a` u dva smjera te nacrtati veći pravokutnik. Ako smo crtanje počeli iz donjeg lijevog kuta, za crtanje stalka potrebno je pomaknuti se udesno za `(8*:a+2*:a)/2-(2*:a)/2=4*:a` i nastaviti crtanje stalka po uputama iz skice.

# Zadatak 6.2 - Signal

Prije početka crtanja pravokutnika, možemo odrediti boju kojom je potrebno popunjavati njihovu unutrašnjost. Kako je zadano u zadatku, ako je broj popunjenih pravokutnika manji od polovice ukupnog broja pravokutnika (`:s<:n/2`), boja za popunjavanje je crvena, pa pozivamo naredbu `SETFC "RED`. Slične provjere obavimo za preostala dva uvjeta za žutu, odnosno zelenu boju.

Crtanje pravokutnika možemo raditi pomoću `FOR` petlje. U svakom koraku nacrtamo pravokutnik širine `:b` i visine `:a+(:i-1)*:c` piksela te se pomaknemo u unutrašnjost pravokutnika s podignutim perom. Ako smo pravokutnik crtali iz donjeg lijevog kuta, možemo se, primjerice, pomaknuti za `:b/2` piksela udesno i 2 piksela prema gore. Tada popunjavamo pravokutnik naredbom `FILL` pod uvjetom da je brojač petlje :i manji ili jednak broju `:s` te se podignuta pera pomičemo na poziciju za crtanje sljedećeg pravokutnika.

# Zadatak 6.3 - Ladice

Za crtanje jedne ladice preporučljivo je napisati potprogram kako bi kod bio čitljiviji. To uključuje crtanje pravokutnika te ručke u sredini ladice. Prvi problem koji je potrebno riješiti jest određivanje visine i širine pojedine ladice. Ako se u nekom retku nalazi `:x` ladica, a širina cijelog ormara iznosi `:a` piksela te razmak između ladica `:c` piksela, onda je širina svake ladice `(:a-(:x+1)*:c)/:x` jer redak sadrži `:x-1` razmak između `:x` ladica te 2 razmaka između ruba ormara i krajnje ladice, što čini ukupno `:x+1` razmak širine `:c`. Slično određujemo i visinu pojedine ladice koja ne ovisi o broju ladica u retku, već o broju redova ladica. Visina svih ladica iznosi `(:b-(:n+1)*:c)/:x`.

Ladice možemo nacrtati korištenjem petlji `FOR/FOREACH` i `REPEAT`. U svakoj iteraciji vanjske `FOR` petlje s brojačem `:i` izračunamo širinu svake ladice u tom retku te petljom `REPEAT` koja se ponavlja `(item :i :l)` puta crtamo ladice. Na kraju iteracije petlje `FOR` pomičemo se na točnu poziciju za crtanje sljedećeg retka.

# Zadatak 6.4 - Neven

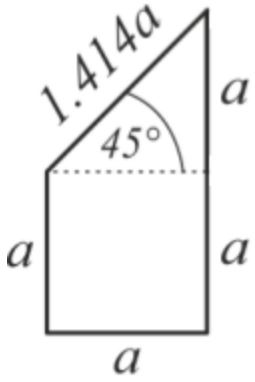
Promotrimo najprije kako iz riječi `:r` izdvojiti podriječ od `:i`-tog do `:j`-tog slova, pri čemu je `:i` manji ili jednak `:j`. Korištenjem naredbe `MAKE` možemo napraviti jednu dodatnu varijablu `:x` koja će na početku biti prazna riječ. Zatim možemo `FOR` petljom proći kroz sva slova od `:i`-tog do `:j`-tog, te svako slovo dodati na kraj riječi `:x` korištenjem naredbe `WORD` koja spaja dvije riječi kako bismo spojili riječ `:x` i trenutno slovo iz riječi `:r`. Ako ovaj postupak ponovimo za sve `:i` i `:j` koji su manji ili jednaki duljini riječi `:r`, izdvojit ćemo sve njezine podriječi.

Nakon što smo izdvojili podriječ, trebamo provjeriti je li ona palindrom. Jedan od mogućih načina da to provjerimo je da s dvije varijable koje nam pamte redne brojeve slova iz riječi prolazimo kroz riječ i redom uspoređujemo slova. Prvo usporedimo prvo i zadnje slovo, zatim jednu varijablu povećamo a drugu smanjimo za 1, te usporedimo drugo i predzadnje slovo i ponovimo taj postupak za sva slova iz riječi. Ako se neka dva uspoređena slova razlikuju, riječ nije palindrom. Kraći način uključuje korištenje naredbe `REVERSE` koja će okrenuti zadanu riječ, pa stoga možemo provjeriti je li riječ `:x` jednaka `(reverse :x)`.

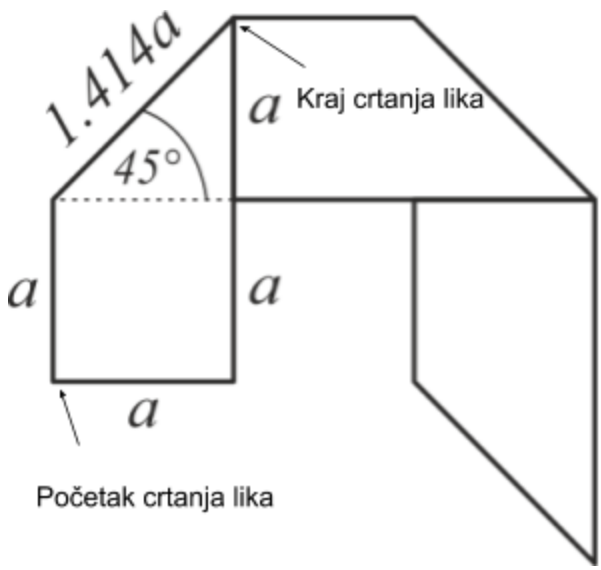
Ako je podriječ koju smo izdvojili palindrom, trebamo je dodati u neku listu koju ćemo na kraju vratiti kao rješenje. Kako bismo bili sigurni da se u toj listi neka riječ neće pojavljivati više puta, prije nego što ubacimo riječ `:x` moramo provjeriti nalazi li se ona već u listi. To možemo učiniti tako da prođemo sve elemente liste i usporedimo ih s riječi `:x`. Ako niti jedan element koji se nalazi u listi nije jednak riječi `:x`, onda ju možemo ubaciti. Kraći način da to napravimo je korištenjem naredbe `MEMBERP` koja za zadanu riječ i listu vraća `TRUE` ili `FALSE` ovisno o tome nalazi li se riječ unutar zadane liste. Alternativno, možemo koristiti i naredbu `REMDUP` koja izbacuje duplikate iz dane liste.

# Zadatak 7.1 - Petlja

Kako bi najlakše riješili zadatak pogledat ćemo skicu danu u zadatku i vidjeti od čega se slika sastoji:



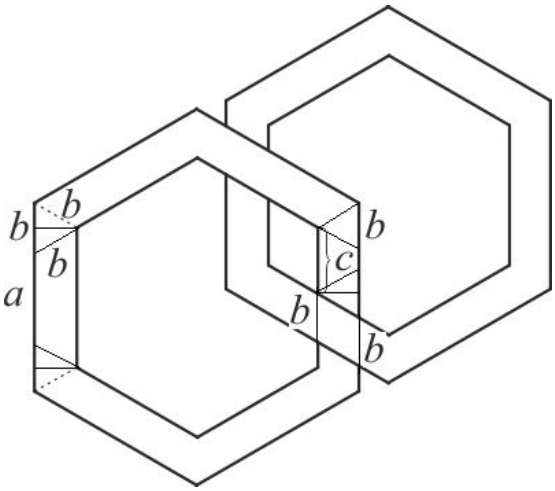

Vidimo da je minimalni element slike kvadrat stranica duljine :a s jednakokračnim trokutom na vrhu. Primjetimo također da imamo dvije verzije istog lika, ovisno na koju stranu je trokut okrenut. Za početak možemo napisati dvije pomoćne metode koje crtaju navedene likove te uz to namjestimo crtanje likova tako da uvijek završimo na vrhu trokuta.




Nakon što to imamo je dovoljno posložiti pozive metoda točnim redoslijedom te još pripaziti ako je potreban manji pomak da se svaki lik krene crtati iz točne pozicije.

Za trećinu bodova se moglo zadatak riješiti bez da se pazi na varijable, ali kod u biti ostaje isti. Zadatak se također može riješiti bez korištenja pomoćnih metoda, ali u svrhu smanjivanja posla je to preporučeno. Isto tako se korištenjem naredbe ponavljanja može kod za crtanje tri lika ponoviti četiri puta da se nacrta cijeli lik što nam dodatno smanjuje posao

# Zadatak 7.2 - Prsten




Zadatak se može riješiti isključivo upotrebom osnovnih naredbi za pomicanje kornjače. Kako je nacrtano na skici, dužinu koja predstavlja razmak između vanjskog i unutarnjeg šesterokuta duljine `:b` možemo nadopuniti do jednakostraničnog trokuta, budući da se radi o pravilnim šesterokutima. Promatranjem skice možemo zaključiti da je stranica unutarnjeg šesterokuta dugačka `:a-:b`, što se također može izračunati upotrebom koordinatne grafike. Sličnim postupkom nadopunjavanjem do jednakostraničnog trokuta saznajemo i duljinu dijela stranice vanjskog šesterokuta lijevog prstena kod preklopa s desnim prstenom.
Također, potrebno je uočiti da su i vertikalni razmaci između šesterokuta koji se ne crtaju (na slici desno ispod oznake za duljinu `:c`) dugački `:b` jer je razmak između najdonjih vrhova vanjskog i unutarnjeg šesterokuta dugačak `:b`.

# Zadatak 7.3 - Meta

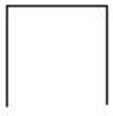
Za rješavanje grafičkog dijela zadatka bilo je potrebno proći kroz obje liste, pozicionirati se na koordinate svake od strijela te nacrtati strijele i to tako da repove i Mirkovih i Slavkovih strijela obojimo crnom bojom. **Nakon toga** bilo je potrebno pozicionirati se u središte ekrana i nacrtati kružnice, primjerice `FOR` petljom koja ide od 1 do `:n`, a crta kružnice radijusa `:r*:i`. Potom ponovno prolazimo kroz Mirkovu i Slavkovu listu, no ovaj puta postavimo boju pera na crvenu, odnosno plavu, i nacrtamo rub repa, a nakon toga postavimo i boju ispune na crvenu, odnosno plavu, i pozicioniramo se unutar repa te ga ispunimo. Time smo riješili problem prolaza kružnica kroz repove strijela.

Za rješavanje tekstualnog dijela zadatka bilo je potrebno pri jednom od prolazaka kroz liste izračunati za svaku strijelu koja je najmanja kružnica u kojoj se nalazi. Za ovaj problem bilo je potrebno poznavati naredbu `INT` koja odbacuje decimalni dio broja (primjerice, `PR INT 9.67` ispisat će 9).
Kada se pozicioniramo na vrh strijele, udaljenost do ishodišta bit će `DISTANCE [0 0]`, a to znači da se ta strijela nalazi unutar `(INT (DISTANCE [0 0])/:r)+1` kružnice po redu, ako kružnice računamo od najmanje prema najvećoj. No, kako je u zadatku zadano da se kružnice računaju od najveće prema najmanjoj, tada se radi o `:n-(INT (DISTANCE [0 0])/:r)`-toj kružnici. Broj bodova koji osvaja strijela je redni broj kružnice*10 pa će tako svaka strijela osvojiti `(:n-(INT (DISTANCE [0 0])/:r))*10` bodova. Bodove je potrebno zbrojiti posebno za Mirka i Slavka i vratiti naredbom `OP`.

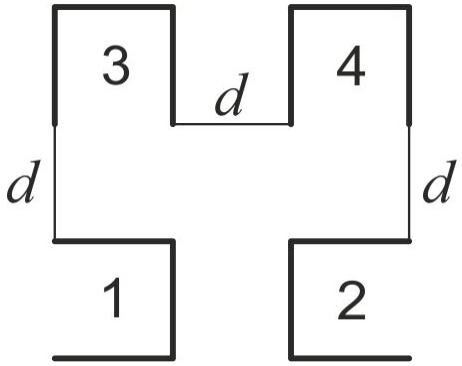
# Zadatak 7.4 - Krivulja

Ovaj zadatak je klasični primjer fraktala (likova koji su sastavljeni sami od sebe u beskonačnost) gdje je zadano prvih par koraka na temelju kojih se može shvatiti pravilnost kojom se fraktal crta. U našem slučaju imamo pojednostavljeni fraktal koji je krivulje te nije beskonačan.

Krenimo prvo od osnovne razine naše krivulje:



Vidimo da se osnovna razina krivulje sastoji od tri crte duljine `:d`. Pogledajmo sad prvi sljedeći korak:




Na skici vidimo da se naša osnovna razina krivulje pojavljuje četiri puta u različitim orijentacijama te su sve razine dodatno razmaknute za `:d`. Možda nije odmah intuitivno jasno zašto, ali ova dva koraka su nam dovoljna da znamo odmah i nacrtati sve ostale korake. Dodatno ćemo si olakšati kodiranje tako što ćemo rekurziju podijeliti na dvije rekurzije, jednu koja crta krivulju slijeva nadesno i jednu koja crta krivulju zdesna nalijevo. Ovo činimo iz razloga koji je vidljiv i sa skice; krivulje 1 i 2 ćemo morati nacrtati zdesna nalijevo, a krivulje 3 i 4 slijeva nadesno.

U nastavku slijedi službeno rješenje s dodanim komentarima:

```
to krivulja :n :d
  rekLD :n :d                    ; Za početak pozivamo rekurziju koja crta krivulju
                                   slijeva nadesno
end

to rekLD :n :d                 ; Ovo je rekurzija koja crta našu krivulju slijeva
                                   nadesno
  if :n = 0 [stop]             ; Uvjet za zaustavljanje, općenito bismo koristili
                                   :n = 1 kao uvjet za zaustavljanje koji crta osnovnu
                                   krivulju, ali nema potrebe za to u ovom zadatku

  rt 90
  rekDL :n - 1 :d              ; Crtamo krivulju označenu s "1" na skici zdesna
                                   nalijevo
  lt 90 fd :d                  ; Možemo se odmah pomaknuti na sljedeću
                                   krivulju jer nas je prethodni poziv pomaknuo na
                                   točnu poziciju
  rekLD :n - 1 :d              ; Crtamo krivulju označenu s "3" na skici
                                   slijeva nadesno
  rt 90 fd :d lt 90            ; Imamo isti efekt kao i u prethodnom rekurzivnom
                                   pozivu, možemo se odmah pomaknuti jer ćemo
                                   po završetku crtanja krivulje niže razine biti na
                                   točnoj poziciji
  rekLD :n - 1 :d              ; Sljedeća dva poziva funkcioniraju na istom
principu
  lt 180 fd :d rt 90
  rekDL :n - 1 :d
  rt 90                        ; Na kraju se još dodatno orijentiramo tako da nam
                                   kornjača pokazuje prema gore, ovo moramo
                                   napraviti jer smo u svim prethodnim pozivima
                                   rekurzije pretpostavili da će kornjača po završetku
                                   crtanja neke krivulje biti tako orijentirana
end

to rekDL :n :d                 ; Ova rekurzija crta našu krivulju zdesna nalijevo, a
                                   kod je simetričan prethodnoj rekurziji
  if :n = 0 [stop]

  lt 90
  rekLD :n - 1 :d
  rt 90 fd :d
  rekDL :n - 1 :d
```

```
  lt 90 fd :d rt 90
  rekDL :n - 1 :d
  lt 180 fd :d lt 90
  rekLD :n - 1 :d
  lt 90
end
```

# Zadatak 8.1 - Čovjek

Ovaj zadatak, kao i većina zadataka, se najlakše rješava tako da napišemo pomoćne metode koje možemo koristiti kako bi si smanjili količinu posla.

Za 40% (12) bodova u zadatku možemo ignorirati varijablu `:r` i praviti se da su svi šesterokuti spojeni. Ovo zadatak čini malo lakšim za kodiranje i nema potrebe da se pazi na podizanje kornjače i ostavljanje praznina. S obzirom da trebamo crtati puno šesterokuta, možemo napisati pomoćnu metodu koja nam crta šesterokut određenih dimenzija. Jedna zanimljiva stvar koju možemo napraviti s tom metodom je dodati jedan dodatni parametar koji nam kaže za koliko stranica će se kornjača pomaknuti nakon crtanja šesterokuta. Konkretno, ako u toj metodi imamo naredbu `REPEAT 6`, nju ćemo pretvoriti tako da piše `REPEAT 6+:p`. Varijabla `:p` će nam smanjiti broj naredbi potrebnih da nam se kornjača lijepo pozicionira za crtanje sljedećeg šesterokuta.

Nakon što smo riješili osnovni problem preostaje nam riješiti problem razmaka između šesterokuta. Srećom, za ovaj problem možemo također napisati jednu pomoćnu funkciju koja nam podigne kornjaču, pomakne je za točan razmak te na kraju spusti. Ovaj kod je dovoljno pozvati svaki put kad smo kornjaču u jednostavnijem zadatku pozicionirali za crtanje sljedećeg šesterokuta, čime smo skupili sve bodove.

# Zadatak 8.2 - Pruga

Prvi korak rješenja zadatka jest nacrtati sve pravokutnike obrubljene crnom bojom i popuniti ih također crnom bojom naredbom `FILL`. Crtanje pravokutnika ponavljamo u petlji `:n` puta tako da se iz središta kružnice podignuta pera pomaknemo unaprijed za `:a+:b+(3*:b)/2` kako bismo došli do središta tračnica te unatrag za `:c/2` i ulijevo za `:d/2` kako bismo došli do donjeg lijevog vrha pravokutnika. Nakon crtanja pravokutnika suprotnom se koracima vraćamo u središte kružnice i okrećemo za `360/:n`. Sljedeći je korak crtanje tračnica koje se sastoje od četiriju kružnica koje crtamo također crnom bojom. Nadalje, ponavljamo crtanje pravoutnika, najprije nekom bojom različitom od crne i bijele, primjerice crvenom, te unutrašnjost pravokutnika popunimo bijelom bojom. Na taj način rješavamo se dijelova kružnica koji se ne smiju vidjeti. Preostaje još samo na rubove pravokutnika nacrtati crnom bojom.

# Zadatak 8.3 - Logo

Promotrimo najprije kako nacrtati odgovarajući dio kružnog vijenca za neko slovo. Veličina kuta koji zatvaraju rubovi područja svakog slova sa središtem kružnica iznosi 360/26 za manji kut, odnosno 25*360/26 za veći kut. Korištenjem naredbe ARC možemo nacrtati kružne lukove polumjera `:r` i `:r+:x` koji prekrivaju kut od 25*360/26 stupnjeva. Nakon što nacrtamo ta dva

kružna luka, potrebno je spojiti njihove krajeve crtom duljine `:x`, te se pozicionirati unutar dobivenog dijela kružnog vijenca, kako bismo ga mogli ispuniti koristeći naredbu `FILL`.

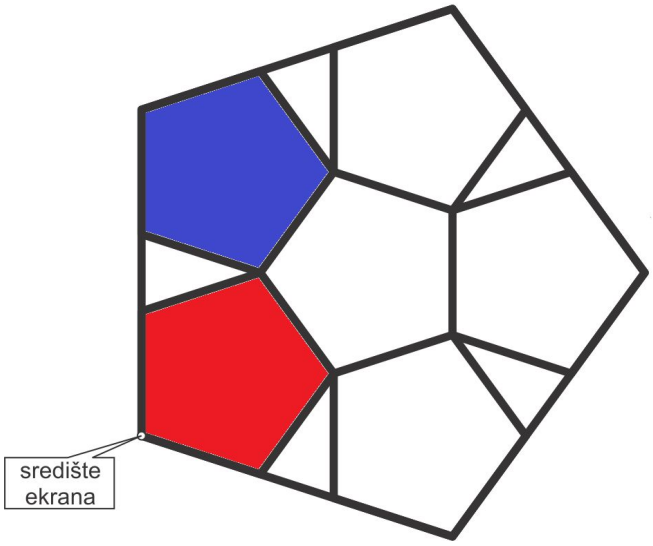
Preostaje još odrediti za koji se kut potrebno okrenuti prije crtanja kružnih lukova, kako bi dobiveni dio kružnog vijenca imao rupu na odgovarajućem mjestu za slovo koje treba predstavljati. Ako svako slovo abecede označimo rednim brojem, počevši od broja 1 za slovo A do broja 26 za slovo Z, jasno je da jedan rub odgovarajućeg dijela kružnog vijenca za slovo s oznakom `:t` zatvara kut od `(360/26)*:t` stupnjeva. Redni broj svakog slova možemo dobiti korištenjem naredbe ASCII koja će nam dati ASCII vrijednost nekog slova. Oduzmemo li tom broju ASCII vrijednost slova A, dobit ćemo njegov redni broj, Opisani postupak potrebno je ponoviti za svako slovo u zadanoj riječi, te povećavati varijablu `:r` nakon svakog nacrtanog dijela kružnog vijenca.

# Zadatak 8.4 - Pahuljica

Za rješavanje ovog zadatka ključno je bilo otkriti u kojem su omjeru stranice manjih peterokuta upisanih u veći peterokut. Kada znamo taj omjer, zadatak se svodi na jednostavnu rekurziju u kojoj do `:n`-te dubine ponavljamo crtanje mnogokuta bez ostavljanja traga i pozivamo rekurziju u svakom vrhu tog peterokuta. U posljednjem pozivu rekurzije potrebno je nacrtati i ispuniti crnom bojom peterokut odgovarajuće duljine stranice.

Promotrimo sada jedan od mogućih načina za određivanje nepoznatog omjera stranica. U zadatku nam je zadana duljina stranice najvećeg peterokuta, a potrebno je onda odrediti duljinu stranice peterokuta koji se nalaze unutar njega. Zbog simetričnosti crteža, vidljivo je da se polovište neke stranice, polovište nasuprotnog vrha toj stranici te zajednički vrhovi manjih peterokuta koji su na toj stranici nalaze na istom pravcu.

Promotrimo sada peterokute sa skice označene crvenom i plavom bojom. Pretpostavimo da je duljina njihove stranice jednaka 100 piksela. Pozicionirajmo se u zajednički vrh tih peterokuta. S obzirom na to da se polovište stranice velikog peterokuta na kojoj leže označeni peterokuti nalazi na istoj visini kao i taj zajednički vrh, korištenjem naredbe ycor dobit ćemo y-koordinatu polovišta te stranice, odnosno polovicu duljine stranice velikog mnogokuta. Budući da nas zanima omjer stranice velikog i manjeg peterokuta, slijedi da je taj omjer jednak `100/(2*ycor)`, odnosno otprilike 0.38196. Primijetite da to što smo u ovom postupku uzeli da je duljina stranice manjeg



peterokuta jednaka 100 piksela nije bilo ograničavanjuće, jer nas je zanimao relativni omjer stranica. Za stranicu manjeg mnogokuta smo mogli uzeti bilokoji broj, te bi traženi omjer ponovno bio jednak.