# VI - I faza projekta

### Globalne promenljive

1. tabla - cuva trenutno stanje igre (ugnjezdene liste, najnizi nivo predstavlja stubove, nivo iznad cuva stubove u jednoj koloni, najvisi nivo cuva kolone)
2. dim - dimenzije table (4 ili 6)
3. isFirstPlayer - informacija ko igra prvi
4. isTwoPlayers - informacija da li je igrac-igrac ili igrac-BOT
5. movesToGo - broj poteza ostalih do zavrsetka igre ( koristi se za proveru da li je igra gotova)
6. nowPlaying - cuva X ili O, u zavisnosti od toga ko je trenutno na poezu

### Funkcije

1. pravilistu - pravi listu od 4 ili 6 “-” elemenata. Sluzi kao pomocna f-ja za pravimatricu.
2. pravimatricu - pravi listu listi (matricu) od 4 ili 6 “-” elemenata zvanjem f-je pravilistu. Sluzi kao pomocna f-ja za pravi3Dmatricu.
3. pravi3Dmatricu – pravi tablu igre (trodimenzionalnu matricu) od 4 ili 6 elemenata zvanjem f-je pravimatricu
4. drawFirstLine - stampa prvu/poslednju liniju (brojeve vrsta/kolona) u zavisnosti od parametra n (4/6)
5. drawTable(lista,n) - stampa trenutno stanje igre koje se nalazi u parametru “lista”, zavisno od parametra n(4/6)
6. dodaj(el lista) - dodaje element u listu(stub), na prvo slobodno mesto (oznaceno simbolom -), ako je lista ne sadrzi element ”-” vraca NIL
7. odigrajpotez – funkcija koja nalazi odredjenu listu listi (matricu) na osnovu zadatog indexa i prosledjuje je funkciji postavikolona koja dalje trazi odredjeni stub na osnovu drugog indexa
8. postavikolona – sluzi kao pomocna funkcija za odigrajpotez funkciju. Sluzi da pronadje odgovarajuci stub u listi stubova i nad njim pozove funkciju dodaj za dodavanje elemenata “x” ili “o”.
9. odigraj – funkcija koa proverava da li znak “-” postoji u stubu na odredjenim koordinatama. Ako postoji poziva se funkcija odigrajpotez, u suprotnom vraca nil.
10. humanPlay - proverava kraj igre, dozvoljava unos poteza u formatu “red kolona”, iscrtava tablu sa novim stanjem, zove funkciju botPlay
11. botPlay - dozvoljava botu ili drugom igracu da odigraju potez u zavisnosti od “isTwoPlayers”, radi sve kao i funkcija humanPlay osium sto na kraju zove humanPlay
12. checkForWinner - trenutno ispisuje “checking for winner” i zavrsava rad programa
13. gameInit - poziva se na pocetku, inicijalizuje igru. Dozvoljava unos svih parametara i podesavanja igre i cuva ih u globalne promenljive (dimenzije table, dva igraca?, ko igra prvi). inicijalizuje tablu, i zove humanPlay ili botPlay u zavisnosti od izbora.

**DRUGA FAZA**

dodate funkcije za odredjivanje pobednika:

1. prebroj(n px po lista) – broji poene u listi atoma I vraca ih u obliku (px po), n prikazuje broj uparenih istih elemenata, uvek krece od 1.
2. countPoints(lista) – sluzi kao wrapper za funkciju prebroj, vraca poene u obliku (px po)
3. racunajStubove(px po lista) – racuna poene u svim stubovima u jednoj koloni, argument lista je lista listi, gde svaka unutrasnja lista predstavlja jedan “stub” na tabli.
4. countAllPillars(px po lista) – racuna poene po stubovima na celoj tabli, koristi funkciju racunaj stubove. Argument lista je lista listi, gde je svaka unutrasnja lista takodje lista listi I predstavlja jednu kolonu.
5. listaPrvih(lista) – izdvaja sve prve elemente ugnjezdenih listi I vraca ih kao listu
6. listaBezPrvih(lista) – vraca listu bez prvih elemenata ugnjezdenih listi
7. kolonaBezD(px po lista) – racuna poene u jednoj koloni, bez racunanja dijagonale I poena u stubovima. Koristi funkcije listaPrvih I listaBezPrvih za odvajanje posebnih listi po kojima se racunaju poeni.
8. getDiag(i glavna lista) – vraca dijagonalu matrice, gde je pevi element dijagonale na poziciji (0, I). U zavisnosti od parametra “glavna”, vraca glavne ili sporedne dijagonale. Argument lista je lista listi (matrica).
9. kolonaD(px po I lista) – racuna poene po dijagonalama u jednoj koloni, argument lista je ta kolona. Poziva se sa i=-2. Radi za matrice velicine 4, 5 ,6.
10. racunajSveKolone – koristi funckije kolonaBezD I kolonaD I sabira njihove vrednosti za sve kolone na tabli, tako da na kraju dobijemo zbir poena po kolonama, racunajuci dijagonale. Argument je lista (cela tabla)
11. racunajSveRedove – poziva funkciju racunajSveKolone, nad transponovanom matricom igre. Tako da na kraju dobijemo ukupne poene po redovima.
12. racunajPoeneBezD(lista) – koristi funkcije za racunanje poena po redovima I kolonama, kao rezultat dobijamo ukupan broj poena na tabli. Bez dijagonala u prostoru.
13. RacunajDijagonale(px po I lista) – radi slicno kao funkcija kolonaD, s tim sto racuna poene dijagonala u prostoru. Pocetno I je uvek -2. radi za matrice velicine 4 I 6.
14. countFinalPoints(lista) – argument je kompletno stanje table igre, koristi gore pomenute funkcije. Izlaz je ukupan broj poena na tabli.
15. odigrajstanja (tabla x y el) – proverava da li je stub na zadatoj lokaciji (x i y) u tabeli popunjen. Ako jeste vraca nil, a ako nije poziva se funkcija odigraj potez.
16. mogucastanja (n) – Dodaje elemente u praznu listu koji se dobijaju pozivom funkcije mogucastanja\_kol. Argument predstavlja najveci index kocke (4 ili 6). Kada index dostigne vrednost -1 funkcija vraca praznu listu na koju se dodaju elementi.
17. mogucastanja\_kol (x y) – Ako odigraj stanja vrati null tj. ako je stub popunjen nastavlja dalje sa rekurzijm, a ako je moguce odigrati potez u tom stubu poziva se funkcija cons koja dodaje elemente koje vraca odigrajpotez u praznu listu koja se dobija nakon izvrsavanja uslova za izlaz iz rekurzije.

**TRECA FAZA**

dodate funkcije:

1. minmax(stanje,dubina a b maxpl) – implementacija algoritma minmax sa alfabeta odsecanjem. A i b predstavljaju alfa i betu maxpl predstavlja da li je na potezu max, dubina je vrednost do koje se ide i stanje je moguci slucaj stanja tabele tokom igre
2. proceni-ver(stanje dim1 dim2), proceni-ver2(stanje dim1 dim2), proceni-ver3(stanje dim1 dim2) - proste funkcije da procenu stanja po vertikali.(Samo radi testiranja).
3. proceni-hor(stanje dim1 dim2), proceni-hor2(stanje dim1 dim2), proceni-hor3(stanje dim1 dim2) - proste funkcije da procenu stanja po horizontali.(Samo radi testiranja).
4. proceni-stanje(stanje dim1 dim2)- fukncija koja sabira vrednosti koje vrate proceni-hor i proceni-ver.
5. jednako(x y z a b c stanje)-funkcija koja proverava da li je element na koordinatama x y z jednak elementu na a b c u datom stanju.
6. ista3(x y z a b c d1 d2 d3 stanje)- funkcija koja proverava da li su 3 elementa na zadatim koordinatama jednaki.
7. ista5(k l pom1 pom2 pom3 pom4 pom5 stanje) funkcija koja proverava da li je 5 prosledjenih elemenata jednako. K i l su indeksi koji se ne menjaju a pom(1,5) je index koji se menja.
8. ista5ver(k l pom1 pom2 pom3 pom4 pom5 stanje) – ista funkcija kao predhodna samo po drugom indeksu ide promena
9. puno(x y stanje)-da li je stub pun na zadatoj lokaciji
10. ima(potez x y z stanje)- da li je potez koji je prosledjen odigran na datoj lokaciji

**CETVRTA FAZA**

dodate funkcije:

1. minimax(stanje,dubina a b maxpl) – prepravljen minmax algoritam sa alfabeta odsecanjem.
2. =dec(x) i =inc(x) – predikati za dekrementiranje i inkrementiranje respektivno.
3. !eq(sign el) – predikat za proveravanje jednakosti sign i el argumenata
4. proveri-stub(lista i j z)- dobija listu koja predstavlja stub na tabli za igru. ako u polju postoji x ili o pravi predikat (on (x ili o) i j z). I,y,z predstavljaju indekse u 3d matrici.
5. proveri-kolona (matrix i j z) – dobija listu stubova(listu polja). Vrsi poziv funkcije proveri-stub za svaki stub u listi.
6. proveri-red (stanje i j z)-dobija stanje tj tablu igre. Vrsi poziv funkcije proveri-kolona za svaku listu stubova.
7. \*T1-RULES\* - pravila za masinu zakljucivanja. Ispituje za 3 u nizu i 4 u nizu.
8. genFacts(talba) – prosledjuje stanje funkciji proveri-red I vrsi njen poziv za generisanje pravila tj \*T1-FACTS\*
9. heuristicsConclusionMachine- pravi promenljivu \*T1-FACTS\* pozivom genFacts funkcije, prebrojava rezultate i akomulira ih u promenljivu acc.
10. proceniStanje(stanje) – koristi se za procenu stanja bez koriscenja masine za zakljucivanje. Koristi funkciju countFinalPoints koja vraca trenutni broj poena X, O. kombinuje njihove poene u jednu vrednost za heuristiku.