# Prva faza projekta

## SJP tim

Stanje igre je predstavljeno na sledeci nacin:

**‘( (tabla) (“XiliO” trenutnoNaPotezu) (brojStekovaX brojStekovaO) velicinaTable )**

Tabla je implementirana kao lista koja sadrzi liste koje predstavljaju redove. Svaki red sadrzi listu polja. TrenutnoNaPotezu—0 ukoliko je na potezu racunar, 1 ukoliko je na potezu covek.

**Opisi funkcija**

1. **praviTablu**—vraca tablu za igranje zadate velicine.
2. **napuniTablu—**postavlja odgovarajuci broj zetona na tablu.
3. **inicijalnoStanje**—generise pocetno stanje (unutar ove fje se odredjuje ko igra prvi).
4. **proveriCiljnoStanje**—provera da li imamo pobednika.
5. **zetoniNaTabli**—vraca polja koja sadrze zetone (pomocna fja kod provere validnosti poteza).
6. **rastojanje** – vraca broj koraka izmedju dva polja na tabli (pomocna fja kod provere validnosti poteza).
7. **meniNajblizi**—vraca polje sa zetonima koje je najblize prosledjenom polju (ili vise ukoliko ih ima, pomocna fja kod provere validnosti poteza).
8. **daLiSePriblizavaNajblizem**—provera da li se prilikom odigravanja poteza priblizivamo najblizem polju koje sadrzi zetone (pomocna fja kod provere validnosti poteza).
9. **crtajMatricu—**generise interfejs na osnovu trenutnog stanja.
10. **validanPotez1—**Proverava da li je se pokusalo da se krene sa belog polja, proverava da li je kretanje po dijagonali I poziva funkciju **proveriValidnoVisinuPriblizavanje** koja vrši ostale provere.
11. **proveriValidnoVisinuPriblizavanje**-poziva fju koja ispituje da li se priblizavamo najblizem za spajanje, ako je ispunjen uslov da se priblizavamo, imamo grananja u odnosu na stanje na tabele, poziv funkcija ako su ispunjeni uslovi, **pomeriStekNaPrazno,** **pomeriStekNaStek.**
12. **pomeriStekNaStek- i**spituje da li su ispunjeni uslovi za spajanje dva steka, ako jesu, poziva funkciju koja **pomeriDiskove** koja vrsi spajanje I promenu stanja.
13. **pomeriDiskove-**pomera diskove sa jednog steka na drugi, ispituje da li stek pun.
14. **izdvojiDiskoveZaPrebacivanje-**izdvaja diskove koje treba prebaciti
15. **izdvojiDiskoveKojiOstaju-**izdvaja diskove koji ostaju na tom polju

# Druga faza

1. **ValidanPotez** -na osnovu prosledjenog stanja I validnog poteza formira novu situaciju na tabli (stanje).
2. **GenerisiMogucaStanja** -poziva pomoćne funkcije koje generišu sva legalna stanja za igrača na potezu.
3. **VratiMogucaStanja**- ide kroz listu pozicija sa kojih takmicar može da odigra, I poziva pomoćnu funkciju **generisiStanjaZaToPolje**, koja generise legalna stanja za konkretno poziciju.
4. **GenerisiStanjaZaToPolje -** Poziva pomoćne funkcije za svako susedno polje po dijagonali , za konkretnu poziciju, pomoćne f-je generišu legalne poteze.
5. **GenerišiStanja1**- Za konkretno polje i jedno od susednih na dijagonali, ispituje sve moguće legalne poteze koji mogu da se odigraju i vraća sva legalna stanja koja mogu da se odigraju između ta dva polja.
6. **PotencijalniMoguciPotezi**-Vraća listu pozicija žetona sa koje je moguće odigrati poteze, prolazi kroz matricu i određuje pozicije.
7. **ProveriPolje**-Proverava da li se može odigrati neki potez sa tog polja, i to su dva slučaja, kada je prvi element u steku, element igrača na potezu i oko njega su prazna polja i drugi slučaj kada se u steku nalazi element igrača na potezu i oko njega postoji barem jedan stek.
8. **ProveriOkoloDaLiJePrazno**-Proverava da li oko tog polja postoji neki stek.
9. **Igraj-**Obezbeđuje igru čoveka i čoveka i proverava da li je došlo do kraja igre.
10. **OdigrajIProveriValidan**-Proverava da li je potez validan, ako jeste prosleđuje vraćeno novo stanje, ako nije upozorava igrača i zahteva unos novog poteza.

# Treca faza

1. **HeuristikaStanja**-vraca vrednost heuristike za prosledjeno stanje.
2. **MaxPotez** –bira najbolji potez za Max igraca .
3. **MinPotez**- bira najbolji potez za Max igraca .
4. **MaxPomPetlja**-pomocna fja u MaxPotez. Realizuje petlju rekurzijom.
5. **MinPomPetlja**-pomocna fja u MinPotez. Realizuje petlju rekurzijom.