**Finalna verzija projekta Byte u Lisp-u**

**Naziv tima: TIME**

**Članovi tima: 1. Milošević Ilija 16213**

**2. Tomović Emilija 16419**

**Funkcije koriscene u projektu**

1. **drawTable-** glavna funkcija koja iscrtava celu tablu, pozivajući pomoćne funkcije
2. **initializeTable-** funkcija koja vrši inicijalizaciju table na osnovu primljenih parametara, koristi podatke sa ulaza
3. **playGame-** funkcija za pocetak igre
4. **playHumanFirst-** omogucava igru gde covek igra prvi, a posle njega igra bot
5. **playBotFirst-** omogucava igru gde bot igra prvi, a posle njega covek
6. **playBot-** funkcija za igru bota(minimax)
7. **createClearList-** izbacuje nevazece poteze iz liste mogucih poteza
8. **chooseTable-** na osnovu unetog podatka o željenoj veličini table, 8 ili 10 poziva funkcije koje će inicijalno da popune tablu, tj.parametar state će dobiti izgled table, koja je formulisana kao graf gde su čvorovi pozicija na tabli sa svim susedima i stanjem stack-a
9. **draw-** funkcija koja poziva funkcije za prikaz table u odnosu na prosledjenu veličinu
10. **numberToLetter –** menja odredjeni broj u slovo
11. **createNeighbourList –** izracunava cvorove susede za zadati cvor, vraca list, flag kad dodje do 2 da onda dodajemo jedan vrsti
12. **createList –** kreira cvor sused
13. **createNode –** kreira cvor koji cine polje, njegovi susedi i stanje na tabeli
14. **stateOfField –** inicijalizacija stanja
15. **fillTable –** saljemo inicijalno za row i column 1 da bi mogao da broji, korak je 2 za kolone, vodi racuna da parne vrste krecu od 2 za kolone
16. **drawUniversalFields –** prikaz table uz pomoc funkcije format(omogucen odabir moda(8x8 ili 10x10)
17. **drawUniversalTable-** kreira tabelu za bilo koju unetu vrednost n
18. **drawFields –** iscrtava polja, i to za svaku vrstu
19. **drawEightRow, drawTenRow –** iscrtava vrste za 8x8/10x10
20. **fillEightRow, fillTenRow-** popunjava vrste za 8x8/10x10
21. **checkEnd-** proverava da li smo došli do kraja, a to je ako 2/3 ili 3/5 pripada jednom igraču ili ako smo došli do kraja sve je popunjeno, ova funkcija vraća samo da li je kraj, ne i ko je pobednik
22. **countX, countO-** prebrojava koliko stack-ova pripada X/O igraču
23. **countStack-** prebrojava visinu stack-a na nekom polju
24. **checkFullStack-** provera da li je stek dosao do visine 8 da bismo upisali u result vrednost ciji je, ko vodi ka pobedi
25. **clearFullStack-** oslobadja ona stanja gde je stek dosao do 8, cisti tabelu
26. **findNode-** pronalazi traženi čvor u grafu
27. **checkIfNodeExists-** proverava da li neki čvor postoji u prosleđenoj listi
28. **checkIfNeighboursHaveStack-** proverava da li susedni čvorovi poseduju neki element na svom polju, vraća true ako postoji, u suprotnom null
29. **checkIfNeighbours-** proverava da li su dva čvora susedi
30. **findInNeighbours-** proverava da li se čvor nalazi u prosleđenoj listi suseda
31. **new-nodes-** smešta nove čvorove potomke koji se već ne nalaze u listi obrađenih čvorova, ova funkcija je potrebna za obilazak po širini
32. **processNode-** proverava da li je neki čvor član obrađenih čvorova, tj.da li se potomak nalazi u listi obrađenih, ako se ne nalazi vraća true, a ako je već obrađen nil
33. **breadthFirst-** graph nam je predstavljen kao stanje tabele, to je parametar koji se prosleđuje za argument graph, ova funkcija vrši obilazak po širini
34. **createProcessed-** kreira listu obrađenih čvorova, tako što na već postojeću listu dodaje novi element
35. **createDescendants-** kreira listu suseda/potomaka za one čvorove koji se ne nalaze u listi obrađenih
36. **findPathInNeighbours-** funkcija koja za susedne čvorove poziva obilazak po širini i kreira listu puteva, do najbližeg čvora čiji stack je veći od 0
37. **findNeighbours-** vraca sve potomke prosledjenog cvora
38. **getPaths-** za prosleđene čvorove kreira putanju, pomoćna funkcija funkcije findPathInNeighbours
39. **findMinLength-** nalazi polje do kog se stiže sa najmanjom dužinom puta
40. **returningMinPath-** vraća listu elemenata sa najmanjom dužinom do tog trenutka
41. **checkIfEndNoteExists–** proverava da li se end cvor nalazi u listi validnih koja se kreira u ovoj fji
42. **checkNodeInValidList–** proverava da li je end cvor u listi validnih
43. **checkNeighbourStack-** proverava ciljni cvor
44. **validateMove–** proverava da li je potez validan
45. **offerNewMove–** nudi novi potez ukoliko je odigran pogresan
46. **checkIfStackStartsWith-** proverava da li stack koji pokusavamo da pomerimo pocinje odredjenim znakom(’X’ ili ’O’)
47. **makeMove–** ukoliko je potez validan odigrava potez
48. **findAppropriateNodes-** vraca cvorove ciji stekovi pocinju sa X ili O u zavisnosti od BW
49. **makeAllMoves-** za svaki cvor koji vrati findAppropriateNodes poziva se makeMove gde je startPoint iz findAppropriateNodes, a endPoint if findNeighbours
50. **createDifferentMoves-** proverava da li moze sa neke visine steka da odigra
51. **createListWithHeights-** kreira poteze sa svih visina
52. **goodNeighbour-** proverava da li je potez dobar
53. **getStackStartingFrom–** vraca stack od elementa n pa do prve tacke
54. **getStackTo–** vraca stack od 0 do n
55. **mergeStacks–** menja stack prosledjenom elementu
56. **changeStackOfElement-** menja stack prosledjenom elementu
57. **evaluateState –** funkcija za procenu stanja, koliko koje vodi do pobede
58. **max-value/min-value –** funkcije za izracunavanje minmaxa sa alfa beta odsecanjem
59. **minimax –** funkcija za odigravanje najboljeg poteza
60. **createFacts-** kreiranje cinjenica
61. **getTopOfStack-** vraca vrh steka
62. **evaluateState-** procena stanja
63. **evaluateStateStack-** procena stanja steka

U izradi projekta pomogao Aleksandar Stoiljkovic