***FAJL checkLogic.py***

def jelSlobodno(a: int, b: int, fields: list[Label]) -> bool:

Funkcija koja proverava da li je polje na koje korisnik želi da stane slobodno, odnosno da li se na njemu ne nalazi već neki od piona. Kao ulazne parametre ima željene koordinate polja i listu koja sadrži sva polja na tabeli. Vraća True ili False.

def dijagonalaGoreDesno(a: int, b: int, zidovi: list[Label]) -> bool :

Funkcija koja proverava da li pion može da se kreće po gornjoj desnoj dijagonali. Kao ulazne parametre sadrži koordinate željenog polja i listu u kojoj se nalaze svi postavljeni zidovi na tabli. Vraća True ili False.

def dijagonalaGoreLevo(a: int, b: int, zidovi: list[Label]) -> bool :

Funkcija koja proverava da li pion može da se kreće po gornjoj levoj dijagonali. Kao ulazne parametre sadrži koordinate željenog polja i listu u kojoj se nalaze svi postavljeni zidovi na tabli. Vraća True ili False.

def dijagonalaDoleDesno(a: int, b: int, zidovi: list[Label]) -> bool :

Funkcija koja proverava da li pion može da se kreće po donjoj desnoj dijagonali. Kao ulazne parametre sadrži koordinate željenog polja i listu u kojoj se nalaze svi postavljeni zidovi na tabli. Vraća True ili False.

def dijagonalaDoleLevo(a: int, b: int, zidovi: list[Label]) -> bool :

Funkcija koja proverava da li pion može da se kreće po donjoj levoj dijagonali. Kao ulazne parametre sadrži koordinate željenog polja i listu u kojoj se nalaze svi postavljeni zidovi na tabli. Vraća True ili False.

def proveraPromenePesaka(a: int, b: int, c:int, args: list[int], zidovi: list[Label], fields: list[Label], pocetak:list[int]) -> bool :

Funkcija koja proverava promenu pešaka, odnosno da li je moguće da se pešak kreće u željenom pravcu, na željeno polje. Kao ulazne parametre sadrži koordinate polja na koja korisnik želi da postavi piona, indeks piona, listu trenutnog stanja (u kojoj se između ostalog nalaze i trenutne koordinate piona), listu svih polja na tabli i listu početnih koordinata svih piona. Vraća True ili False.

def proveraPromeneZida(args: list[int], fields: list[Label], zidovi: list[Label], a: list[int], walls: list[Label], frame: Frame) -> bool :

Funkcija koja proverava da li je moguće da se zid postavi na željeno mesto. Kao ulazne parametre ima listu polja između kojih se zid postavlja, listu svih polja na tabli, listu svih postavljenih zidova, listu sa upamćenim početnim parametrima igre, listu svih labela na koje je moguće postaviti zidove i frame. Vraća True ili False.

def imaLiZidova(a: int, b: int, args: list[int]) -> bool:

Funkcija koja proverava da li je igraču ostalo zidova na raspolaganju. Kao ulazne parametre ima boju zida, indeks igrača i listu trenutnog stanja. Vraća True ili False.

def postojiPutanja(a: list[int], zidovi: list[Label], fields: list[Label]) -> bool:

Funkcija koja implementira algoritam za traženje. Kao ulazne parametre sadrži listu koja čuva početne parametre igre, listu svih postavljenih zidova na tabli i listu svih polja na tabli. Proverava da li od piona postoji put do početnog polja protivničkog igrača. Vraća True ili False.

def napraviGraf(n, m: int, fields: list[Label], zidovi: list[Label]) -> dict[str, list[str]]:

Funkcija koja pravi graf od svih polja na tabli. Kao ulazne parametre ima n i m koordinate table, sva polja table, kao i sve postavljene zidove na tabli. Pravi graf svih dostupnih (nezagrađenih) suseda svakog polja i vraća ga u obliku heš tablice.

def BreadthFirstSearch(graph: dict[str, list[str]], start: str, end: str, obradjeni: list[str], red: list[str]) -> bool:

Algoritam za traženje. Prolazi kroz graf koji je napravljen u funkciji napraviGraf.

def condition(x: Label, i: int, j: int): return int(x.grid\_info()['row'] - 1) == i and int(x.grid\_info()['column']) == j

def condition2(x: Label, i: int, j: int): return int(x.grid\_info()['row']) == i and int(x.grid\_info()['column']) == j

***FAJL drawing***

def iscrtajTablu(n: int, m: int, frame: Frame, fields: list[Label], walls: list[Label], p11: int, p12: int, p21: int, p22: int, p31: int, p32: int, p41: int, p42: int):

Funkcija koja iscrtava tablu na ekranu. Prosleđuju joj se veličina table, frame, lista polja na tabli, lista labela na koje se mogu postaviti zidovi, početne pozicije sva 4 piona.

def iscrtajIzbor(root: Tk, options: Frame, frame: Frame, btnTypeOfGame: Button, ts0: OptionMenu, ts1: OptionMenu, ts2: OptionMenu, ts4: OptionMenu, tableSizeN: StringVar, tableSizeM: StringVar, numberOfWalls: StringVar, typeOfGame: StringVar, fields: list[Label], walls: list[Label], zidovi: list[Label], a: list[int], b: list[int]):

Funkcija koja iscrtava dijalog za korisnika u kome on bira mod igre, a nakon toga ko igra prvi i unosi početne pozicije pešaka.

def iscrtajZid(args: list[int], frame: Frame, walls: list[Label], zidovi: list[Label], b: list[int]):

Funkcija koja iscrtava zid u određene labele nakon odigranog postavljanja.

def iscrtajPesaka(args: list[int], fields: list[Label], b: list[int]):

Funkcija koja iscrtava pešaka u određenu labelu nakon odigranog pomeranja.

def upamtiPocetnoStanje(fields: list[Label], walls: list[Label], zidovi: list[Label], options: Frame, tableSizeN: StringVar, tableSizeM: StringVar, numberOfWalls: StringVar,

firstPlayer: StringVar, frame: Frame, inPos1: Entry, inPos2: Entry, label1: Label, label2: Label, ts0: OptionMenu, ts1: OptionMenu,

ts2: OptionMenu, ts3: OptionMenu, buttonStart: Button, a: list[int], b:list[int]):

Funkcija koja pamti početne parametre igre nakon što ih korisnik unese. Pamti veličinu table, početne pozicije svih piona, broj zidova, ko igra prvi, ukoliko je odabran mod igre čovek vs. računar.

def upamtiPocetnoStanje2(fields: list[Label], walls: list[Label], zidovi: list[Label], options: Frame, tableSizeN: StringVar, tableSizeM: StringVar, numberOfWalls: StringVar, frame: Frame, inPos1: Entry, inPos2: Entry, inPos3: Entry, inPos4: Entry, label1: Label, label2: Label, label3: Label, label4: Label, ts0: OptionMenu, ts1: OptionMenu,

ts2: OptionMenu, buttonStart: Button, a: list[int], b:list[int]):

Funkcija koja pamti početne parametre igre nakon što ih korisnik unese. Pamti veličinu table, početne pozicije svih piona, broj zidova, ko igra prvi, ukoliko je odabran mod igre čovek vs. čovek.

def izborPocetnihParametara(root: Tk, frame: Frame, a: list[int], b: list[int], fields: list[Label], walls: list[Label], zidovi: list[Label]):

Funkcija koja iscratava dijalog u kome igrač bira početne postavke igre (veličinu table, broj zidova, tip igre).

def preuzimanjePromena(frame: Frame, entryIndex: Entry, entryCoor: Entry, labelMove: Label, a: list[int], b: list[int], fields: list[Label],

walls: list[Label], zidovi: list[Label], buttonStart: Button):

Funkcija koja preuzima promene u toku igre nakon što ih korisnik unese. Poziva funkcije koje proveravaju da li je moguće promeniti trenutno stanje na način na koji je igrač to želeo. Ukoliko je moguće, iscrtava novo stanje, ukoliko nije, vraća poruku korisniku da unese validno novo stanje.

***FAJL state***

def krajnjeStanje(trenutnePozicije: list[int], pocetnePozicije: list[int]) -> tuple[bool, int]:

Funkcija koja proverava krajnje stanje igre. Kao ulazne parametre ima trenutne pozicije piona, kao i njihove početne pozicije. Proverava da li je neki od piona na početnim pozicijama piona drugog igrača. Vraća True ili False, kao i indeks igrača koji je pobedio ukoliko je bool vrednost True. Ukoliko nije kraj igre, vraća False i 0.