

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

3Η ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

Ευάγγελος Μπενέτος, ΑΜ:1072628

A.

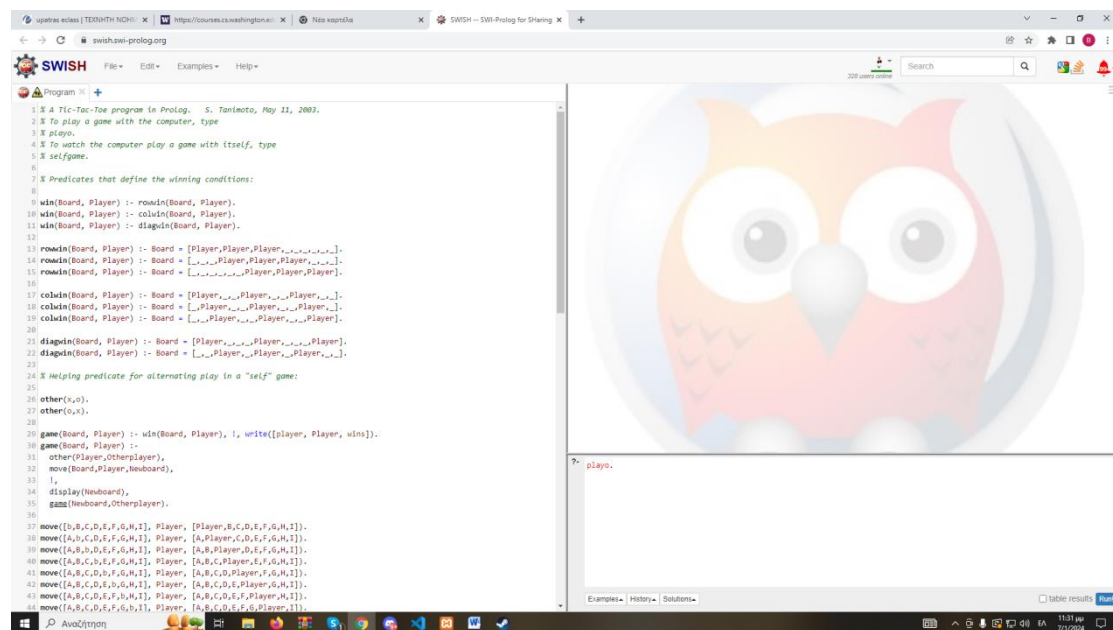
Δίνεται το παρακάτω πρόγραμμα που υλοποιεί το παιχνίδι «τρίλιζα»:

<https://courses.cs.washington.edu/courses/cse341/03sp/slides/PrologEx/tictactoe.pl.txt>

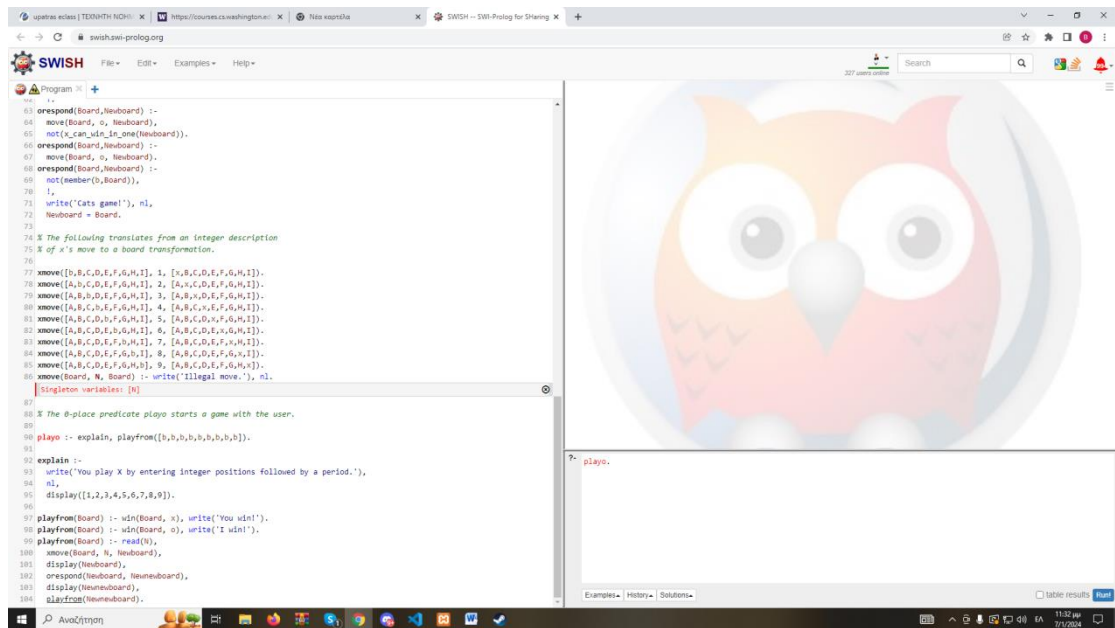
Κατεβάστε το και δοκιμάστε το στον ΗΥ σας. Εξηγείστε σύντομα τη λειτουργία του κατηγορήματος *respond* και τη λειτουργία του προγράμματος.

ΑΠΑΝΤΗΣΗ Α. ΕΡΩΤΗΜΑΤΟΣ:

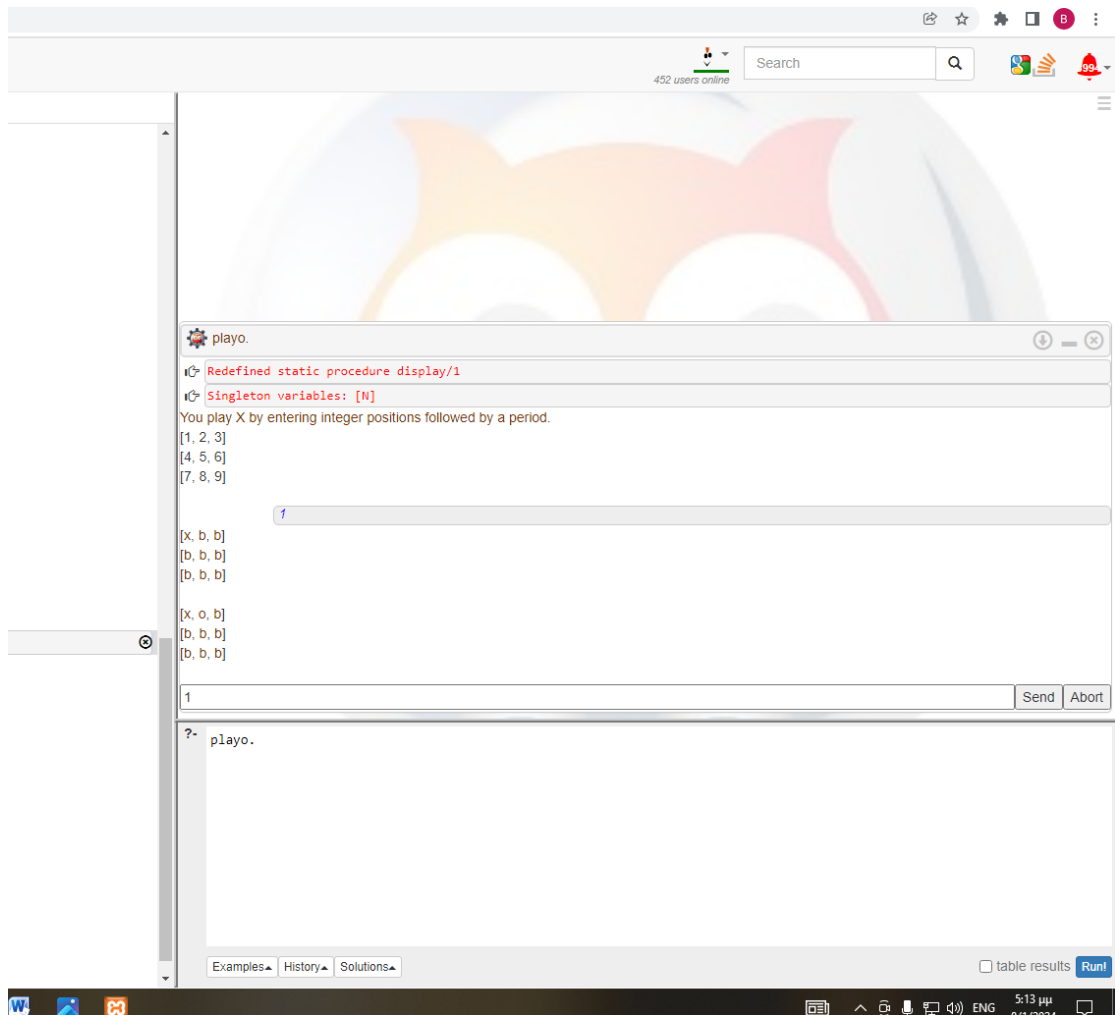
Αρχικά αυτό που έκανα ήταν να τρέξω τον κώδικα από το λίνκ ώστε να μπορώ να πειραματιστώ μαζί του και να παίξω το παιχνίδι της τρίλιζας. Παρακάτω θα παραθέσω με σκρεεν σοτς τον κώδικα εκεί που τον έτρεξα και τα αποτελέσματα που πήρα.



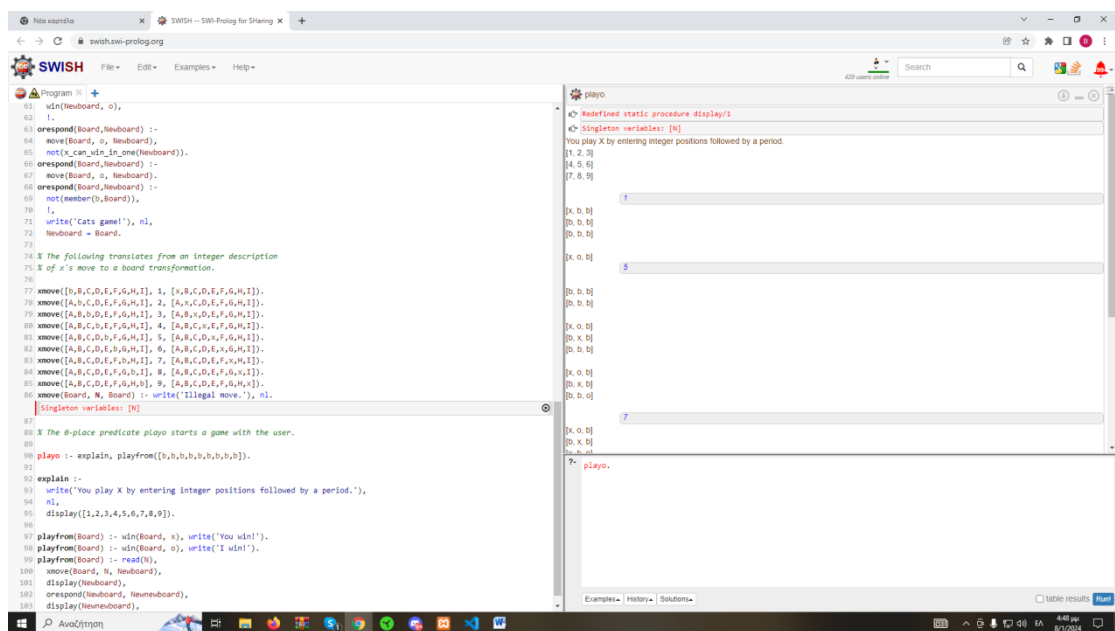
```
1 % A Tic-Tac-Toe program in Prolog. S. Tammeto, May 11, 2003.
2 % To play a game with the computer, type
3 % play.
4 % To watch the computer play a game with itself, type
5 % selfgame.
6
7 % Predicates that define the winning conditions:
8
9 win(Board, Player) :- rowwin(Board, Player).
10 win(Board, Player) :- colwin(Board, Player).
11 win(Board, Player) :- diagwin(Board, Player).
12
13 rowwin(Board, Player) :- Board = [Player,Player,Player,_,_,_,_,_,_].
14 rowwin(Board, Player) :- Board = [_,_,Player,Player,Player,_,_,_,_].
15 rowwin(Board, Player) :- Board = [_,_,_,Player,Player,Player,_,_,_].
16
17 colwin(Board, Player) :- Board = [Player,_,Player,_,Player,_,_,_].
18 colwin(Board, Player) :- Board = [_,Player,_,Player,_,Player,_,_].
19 colwin(Board, Player) :- Board = [_,_,Player,_,Player,_,Player,_,_].
20
21 diagwin(Board, Player) :- Board = [Player,_,_,Player,_,_,_,_].
22 diagwin(Board, Player) :- Board = [_,Player,_,_,Player,_,_,_].
23
24 % Helping predicate for alternating play in a "self" game:
25
26 other(x,_).
27 other(_,x).
28
29 game(Board, Player) :- win(Board, Player), !, write([player, Player, win]).
30 game(Board, Player) :-
31   other(Player, OtherPlayer),
32   move(Board, Player, Newboard),
33   !,
34   display(Newboard),
35   game(Newboard, OtherPlayer).
36
37 move([_,_,_,_,_,_,_,_,_], Player, [Player,_,_,_,_,_,_,_,_]).
38 move([_,_,_,_,_,_,_,_,_], Player, [_,Player,_,_,_,_,_,_,_]).
39 move([_,_,_,_,_,_,_,_,_], Player, [_,_,Player,_,_,_,_,_,_]).
40 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,Player,_,_,_,_,_]).
41 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,_,Player,_,_,_,_]).
42 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,_,_,Player,_,_,_]).
43 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,_,_,_,Player,_,_]).
44 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,_,_,_,_,Player,_,_]).
45 move([_,_,_,_,_,_,_,_,_], Player, [_,_,_,_,_,_,_,_,Player,_,_]).
```

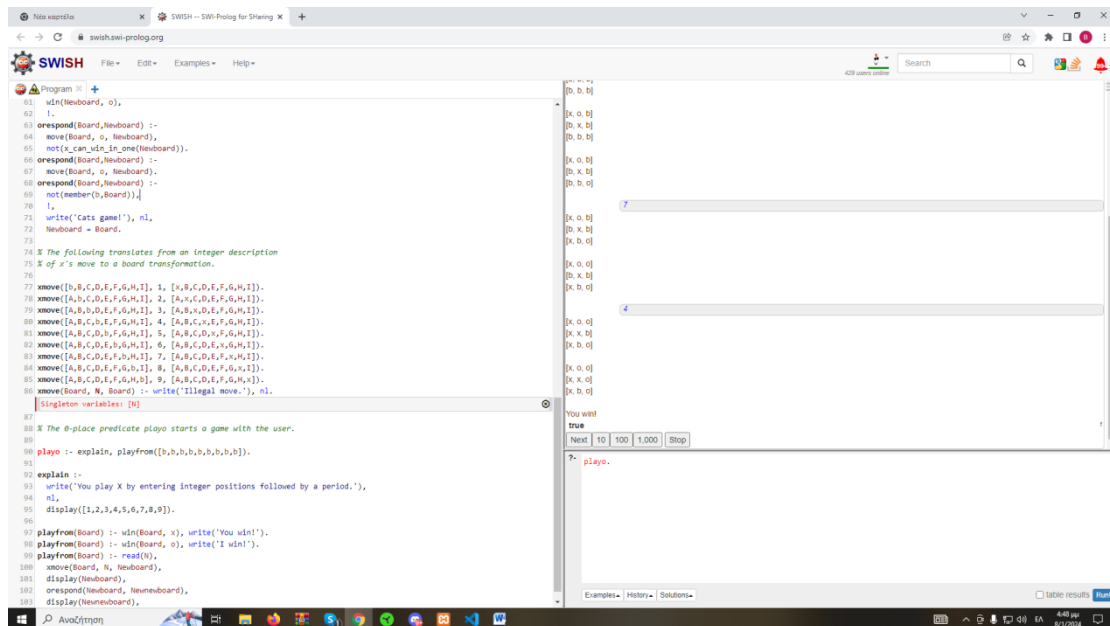


Με την εντολή `playo`. που φαίνεται στα δεξιά στα σκρεεν σοτς είναι ουσιαστικά η εντολή με την οποία παίζει το παιχνίδι ο χρήστης εγώ δηλαδή με λίγα λόγια και θα δείτε τα νούμερα που έβαλα ώστε να προσπαθήσω να νικήσω ή να νικήσει ο υπολογιστής εμένα σύμφωνα με τις εντολές του αλγορίθμου που εκτελεί στην τρίλιζα.



Στο συγκεκριμένο screen shot δείχνω ότι με την χρήση της εντολής playo. έβαλα το 1 και πάτησα send και μπήκε στην θέση που έπρεπε και έπειτα ο υπολογιστής η μηχανή με άλλα λόγια σύμφωνα με τον αλγόριθμο που ακολουθεί έβαλε και αυτή το δικό της σύμβολο.





Λοιπόν στην συγκεκριμένη περίπτωση σύμφωνα με τις θέσεις που επέλεξα να να τοποθετήσω το σύμβολο μου διότι για να το διευκρινήσουμε εγώ σαν χρήστης έχω το X σαν σύμβολο στην τρίλιζα και ο υπολογιστής έχει το O. Αυτό που πρέπει να πούμε είναι ότι οι θέσεις της τρίλιζας ώστε να δηλώσω εγώ κάθε φορά την θέση που θέλω να μπει το σύμβολο μου είναι κάπως έτσι

| | | |
|---|---|---|
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Και κάθε φορά εγώ δίνω μία θέση στον πίνακα και ο υπολογιστής που θεωρείται ο αντίπαλος μου προσπαθεί να εισάγει αφού έχω παίξει εγώ πρώτος να βάλει και αυτός το σύμβολο του ώστε να με κερδίσει.

Στα συγκεκριμένα παραδείγματα που έστειλα με screen shots κατάφερα και έβαλα με τέτοια σειρά τις θέσεις όπου έπρεπε να μπει το σύμβολο μου και κατάφερα και κέρδησα και για αυτό βγήκε το μήνυμα YOU WIN!

Τώρα υπάρχει και ένας άλλος τρόπος ώστε να παιχτεί το παιχνίδι της τρίλιζας και αυτός είναι χωρίς να συμμετέχω εγώ δηλαδή χωρίς να συμμετέχει κάποιος χρήστης και απλά να παίζει μόνος του ο υπολογιστής. Αυτό γίνεται με χρήση μίας άλλης εντολής της selfgame. και παρακάτω θα δείξω με screen shots και φένεται ξεκάθαρα ότι παίζει μόνος του ο υπολογιστής και δεν συμμετέχω πουθενά εγώ.

441 users online

Search

selfgame.

Redefined static procedure display/1

Singleton variables: [N]

[x, b, b]
[b, b, b]
[b, b, b]

[x, o, b]
[b, b, b]
[b, b, b]

[x, o, x]
[b, b, b]
[b, b, b]

[x, o, x]
[o, b, b]
[b, b, b]

[x, o, x]
[o, x, b]
[b, b, b]

[x, o, x]
[o, x, o]
[b, b, b]

[x, o, x]
[o, x, o]
[x, b, b]

?- selfgame.

441 users online

Search

[o, o, b]
[x, o, b]
[b, b, b]
[b, b, b]

[x, o, x]
[b, b, b]
[b, b, b]

[x, o, x]
[o, b, b]
[b, b, b]

[x, o, x]
[o, x, b]
[b, b, b]

[x, o, x]
[o, x, o]
[b, b, b]

[x, o, x]
[o, x, o]
[x, b, b]

[x, o, x]
[o, x, o]
[x, o, b]

[player, x, wins]
true

?- selfgame.

Από το μήνιμα φαίνεται ότι κέρδησε ο παίκτης με το σύμβολο X, θεωρητικά πάντα διότι όπως είπαμε ο υπολογιστής μόνος του έπαιξε σύμφωνα με την χρήση της εντολής selfgame.

Τώρα όσον αναφορά την λειτουργία του κατηγορήματος respond αυτό που κάνει ουσιαστικά είναι να επιστρέφει την καλύτερη δυνατή κίνηση ή θέση όπως θέλουμε μπορούμε να το πούμε για το σύμβολο που διαχειρίζεται ο υπολογιστής το O με βάσει πάντα το πώς λειτουργεί το παιχνίδι της τρίλιζας. Έτσι προσπαθεί να κερδίσει να δημιουργήσει δηλαδή με αυτό το σύμβολο τρίλιζα ή προσπαθεί να μην αφήσει το άλλο σύμβολο X να κερδίσει αυτό.

B.

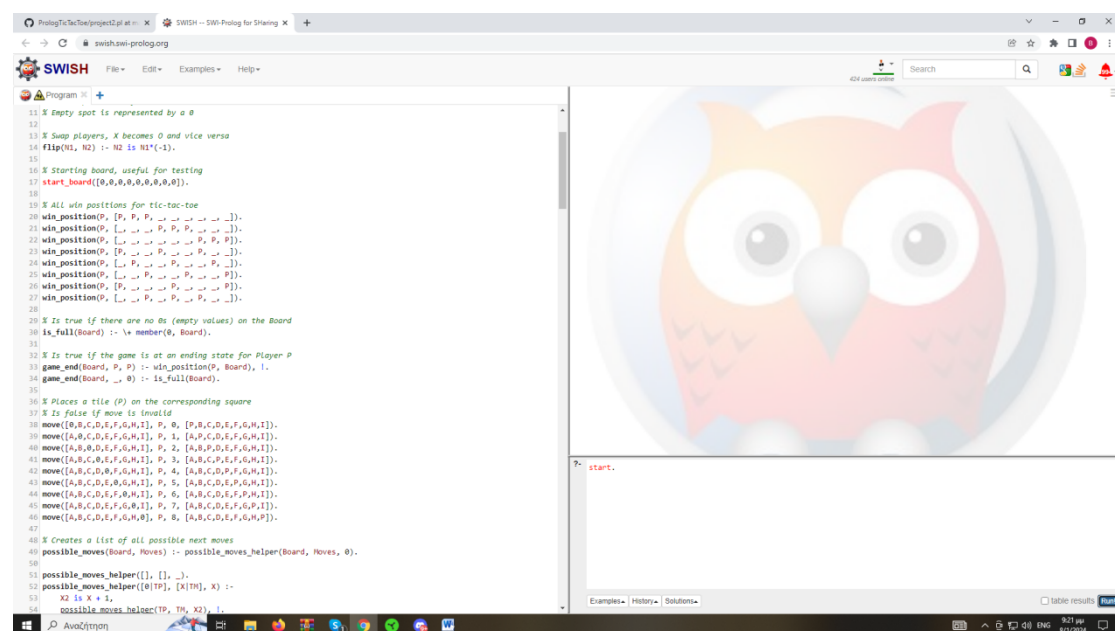
Δίνεται το παρακάτω πρόγραμμα που υλοποιεί το παιχνίδι «τρίλιζα»:

<https://wiki.ubc.ca/Course:CPSC312-2021/Tic-Tac-Toe>

Κατεβάστε το και δοκιμάστε το στον ΗΥ σας. Εξηγείστε σύντομα τη βασική διαφορά της λειτουργίας του από το πρόγραμμα του μέρους Α.

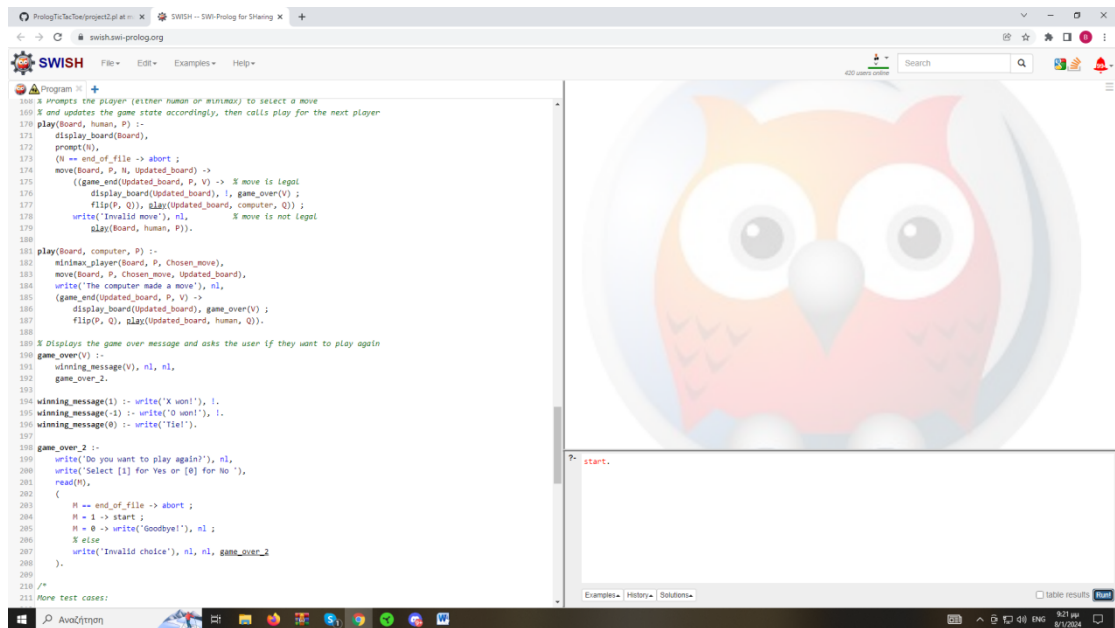
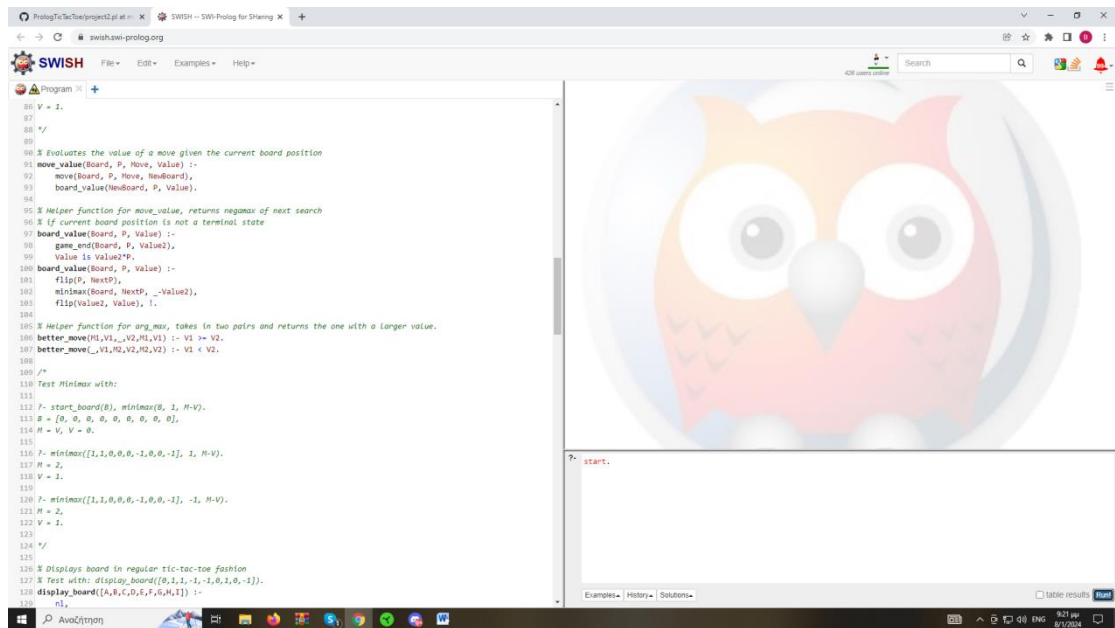
ΑΠΑΝΤΗΣΗ Β. ΕΡΩΤΗΜΑΤΟΣ:

Αυτό που έκανα και σε αυτό το ερώτημα όπως και στο προηγούμενο είναι να πάρω τον κώδικα και να τον τρέξω ώστε να αντιληφθώ τι κάνει. Οπότε όπως και πριν θα σας δείξω που έτρεξα και αυτόν τον κώδικα με κάποια screen shots.



The screenshot shows a web browser window with the URL <https://wiki.ubc.ca/Course:CPSC312-2021/Tic-Tac-Toe>. The browser displays a SWISH program editor interface. The editor shows a list of SWISH commands for a Tic-Tac-Toe game, including functions for win conditions, game state, and moves. A large, stylized owl illustration is visible in the background of the editor. The bottom of the browser window shows the Windows taskbar with various application icons and the system clock indicating 9:21 AM on 5/1/2024.

```
11 X Empty spot is represented by a 0
12
13 X Swap players, X becomes O and vice versa
14 flip(N1, N2) :- N2 is N1*(-1).
15
16 X Starting board, useful for testing
17 start_board([0,0,0,0,0,0,0,0,0]).
18
19 X All win positions for tic-tac-toe
20 win_position(P, [P, P, P, _, _, _, _, _, _]).
21 win_position(P, [_, P, P, P, _, _, _, _, _]).
22 win_position(P, [_, _, P, P, P, _, _, _, _]).
23 win_position(P, [_, _, _, P, P, P, _, _, _]).
24 win_position(P, [_, _, _, _, P, P, P, _, _]).
25 win_position(P, [_, _, _, _, _, P, P, P, _]).
26 win_position(P, [_, _, _, _, _, _, P, P, P]).
27 win_position(P, [_, _, P, _, P, _, P, _, _]).
28
29 X Is true if there are no 0s (empty values) on the Board
30 is_full(Board) :- \+ member(0, Board).
31
32 X Is true if the game is at an ending state for Player P
33 game_end(Board, P, 0) :- win_position(P, Board), 1.
34 game_end(Board, _, 0) :- is_full(Board).
35
36 X Places a tile (P) on the corresponding square
37 X Is false if move is invalid
38 move([0,0,0,0,0,0,0,0,0], P, 0, [P,0,0,0,0,0,0,0,0]).
39 move([0,0,0,0,0,0,0,0,0], P, 1, [0,P,0,0,0,0,0,0,0]).
40 move([0,0,0,0,0,0,0,0,0], P, 2, [0,0,P,0,0,0,0,0,0]).
41 move([0,0,0,0,0,0,0,0,0], P, 3, [0,0,0,P,0,0,0,0,0]).
42 move([0,0,0,0,0,0,0,0,0], P, 4, [0,0,0,0,P,0,0,0,0]).
43 move([0,0,0,0,0,0,0,0,0], P, 5, [0,0,0,0,0,P,0,0,0]).
44 move([0,0,0,0,0,0,0,0,0], P, 6, [0,0,0,0,0,0,P,0,0]).
45 move([0,0,0,0,0,0,0,0,0], P, 7, [0,0,0,0,0,0,0,P,0]).
46 move([0,0,0,0,0,0,0,0,0], P, 8, [0,0,0,0,0,0,0,0,P]).
47
48 X Creates a list of all possible next moves
49 possible_moves(Board, Moves) :- possible_moves_helper(Board, Moves, 0).
50
51 possible_moves_helper([], [], _).
52 possible_moves_helper([0|TP], [X|TH], X) :-
53   X2 is X + 1,
54   possible_moves_helper(TP, TH, X2).
```



Όπως θα δείτε και στα screen shots μία πρώτη διαφορά είναι ότι πέραν ότι προφανώς μιλάμε για διαφορετικό κώδικα η εντολή που θα πατήσουμε run ώστε να μπορούμε να τρέξουμε και παράλληλα να παίξουμε το παιχνίδι είναι η start. σε σχέση με την playo. που χρησιμοποιούσαμε πριν.

Τώρα θα παραθέσω κάποια screen shots παίζοντας το παιχνίδι της τρίλιζας απέναντι στον υπολογιστή-μηχανή που ακολουθεί τα βήματα και τις εντολές του αλγορίθμου και έπειτα θα κρίνουμε και την βασική διαφορά με αυτόν του Α ερωτήματος

415 users online

Search

start.

Redefined static procedure display/1

Select [0] to choose X, or [1] to choose O. End your choice with a period:

0

||

||

||

||

Select an integer from 0 to 8 inclusive. End your choice with a period

0

The computer made a move

X||

|O|

||

Select an integer from 0 to 8 inclusive. End your choice with a period

2

The computer made a move

X|O|X

|O|

||

Select an integer from 0 to 8 inclusive. End your choice with a period

6

The computer made a move

?- start.

Examples History Solutions

table results Run

9:30 μs
8/11/2024

415 users online

Search

X|O|X

|O|

||

Select an integer from 0 to 8 inclusive. End your choice with a period

6

The computer made a move

X|O|X

O|O|

X||

Select an integer from 0 to 8 inclusive. End your choice with a period

5

The computer made a move

X|O|X

O|O|X

X|O|

O won!

Do you want to play again?
Select [1] for Yes or [0] for No

0

Goodbye!

true

0.331 seconds cpu time

Next 10 100 1,000 Stop

?- start.

Examples History Solutions

table results Run!

9:31 μs
8/11/2024

Αρχικά πρέπει να πούμε κάτι το οποίο αλλάζει σε σχέση με το Α ερώτημα είναι ότι τώρα οι θέσεις τις τρίλιζας έχουν αυτήν την αρίθμηση ώστε να μπορείς να επιλέξεις θέση για να βάλεις το σύμβολο σου.

| | | |
|---|---|---|
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

Η κύρια όμως διαφορά σε σχέση με το προηγούμενο ερώτημα και όπως φάνηκε και από τα screen shots καθώς έχασα είναι ότι παίζοντας πολλές φορές και βάζοντας σε πολλές διαφορετικές θέσεις το σύμβολο μου δεν κατάφερα να κερδίσω σε σχέση με το Α ερώτημα όπου κατάφερα να κερδίσω και σχετικά εύκολα κιόλας. Αυτό μετά από αρκετό ψάξιμο στον κώδικα αντιλήφθηκα ότι γίνεται χάρη στην συνάρτηση που χρησιμοποιεί ο δεύτερος αλγόριθμος MINIMAX. Η συγκεκριμένη συνάρτηση είναι κατάλληλη για παιχνίδια στρατηγικής όπως και το δικό μας της τρίλιζας, πρόκειται για μία συνάρτηση που λειτουργεί αναδρομικά και οι οποία εξετάζει κάθε μα κάθε φορά τις πιθανές κινήσεις που μπορεί να κάνει κάθε παίκτης έπειτα αξιολογεί αυτές τις κινήσεις και προσπαθεί να βρεί την καλύτερη δυνατή κίνηση για τον παίκτη, προσπαθεί να βρεί με λίγα λόγια την κίνηση με το μικρότερο κόστος λάθους. Για αυτό τον λόγο αυτή είναι η κύρια διαφορά με το Α ερώτημα και στο οποίο μπορούσαμε εύκολα να κερδήσουμε καθώς για τις κινήσεις χρησιμοποιούσε μια πιο απλή συνάρτηση την `orespond` και έτσι εξηγείται που με τίποτα δεν μπορέσαμε να κερδήσουμε απέναντι στον δεύτερο αλγόριθμο.