جامعة دمشق

Principles of Artificial Intelligence

Tic_tac _Logic with Prolog

يونيو 2023

أشراف :مهندسون العملي

مُقدِّمة من

أمين النصيرات
براءة دلي حسن
أنس الهيمد
وائل دردر
آية موسى

Game Tac_Toc_logic

- هي لعبة مطور من اللعبة الشهيرة Tic_Tac_Toe, وتعرف بأنها لعبة ألغاز تتكون من لوحة مربعة مقسومة الى خلايا صغيرة, وتحتوي على بعض الخلايا التي يتم تحديد مسبقا" برموز (O,X), الهدف من اللعبة هو ملء الخلايا الفارغة بأحد الرمزين (X,O) بحيث يتم احترام القواعد التالية:
- لا يوجد أكثر من رمزين X او O متتاليين في سطر أو عمود .
. عدد X هو نفسه عدد O في كل سطر وعمود.
- جميع السطور وجميع الأعمدة فريدة أي أنه لا تتكرر نفس الرموز فيها بنفس الترتيب .

خوارزمية اللعبة :

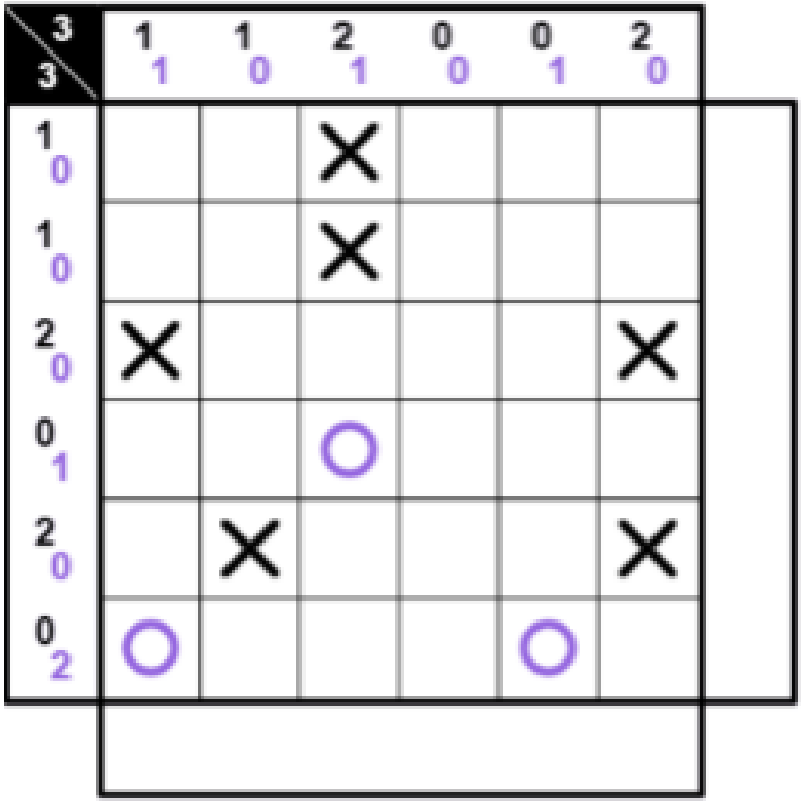
هنا سنشرح الآلية المتبعة في برمجة اللعبة والأجرائيات المستخدمة وطرق التفكير المتبعة : سنبدأ بالحائق والأجرائيات ,من ثم طريقة استخدامها من أجل اختبار صحة الحل من عدمه

1. **في البداية نقوم بتعريف حجم الرقعة:**

```
size(6,6).
```

2.من ثم قمنا بتمثيل الخلايا والرموز التي تحويهاعن طرق اسناديات ,بحيث تعبر عن علاقة بين ثلاث معاملات :السطر والعمود والرمز , بحيث يكون لدينا حقيقة من أجل كل خلية تحتوي رمزا" ضمن الجدول. وفرضنا وجود حقائق ثابتة تمثل الرموز الثابتة في خلايا الرقعة التي لا يستطيع اللاعب تبديلها .

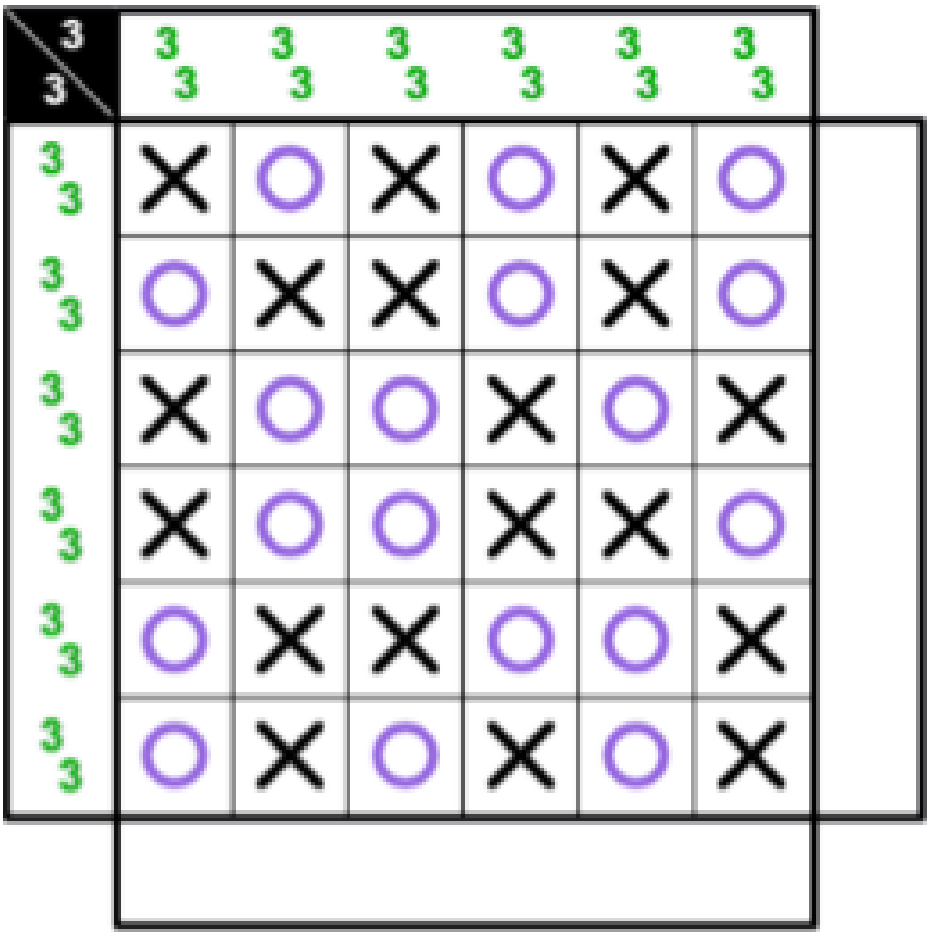
```
fixed_cell(5, 6, x).

fixed_cell(6, 1, o).

fixed_cell(6, 5, o).

fixed_cell(3, 6, x).

fixed_cell(4, 3, o).

fixed_cell(5, 2, x).

fixed_cell(1, 3, x).

fixed_cell(2, 3, x).

fixed_cell(3, 1, x).
```



3.بعدها قمنا بتمثيل خلايا الحل الذي اللاعب سوف يقوم بملئها حسب رغبته باحد الرمزين(X,O),لكن في هذا الجزء من المسألة اعتبرناها كحقائق ثابتة حتى نختبر صحة الاجرائيات المستخدمة في تحقيقي قواعد اللعبة :

خلايا احد الحلول الصحيحة الممكنة للعبة:

```
solve_cell(1, 1, x).
solve_cell(1, 2, o).
solve_cell(1, 4, o).
solve_cell(1, 5, x).
solve_cell(1, 6, o).
solve_cell(2, 1, o).
solve_cell(2, 2, x).
solve_cell(2, 4, o).
solve_cell(2, 5, x).
solve_cell(2, 6, o).
solve_cell(3, 2, o).
solve_cell(3, 3, o).
solve_cell(3, 4, x).
solve_cell(3, 5, o).
solve_cell(4, 1, x).
solve_cell(4, 2, o).
solve_cell(4, 4, x).
solve_cell(4, 5, x).
solve_cell(4, 6, o).
solve_cell(5, 1, o).
solve_cell(5, 3, x).
solve_cell(5, 4, o).
solve_cell(5, 5, o).
solve_cell(6, 2, x).
solve_cell(6, 3, o).
solve_cell(6, 4, x).
solve_cell(6, 6, x).
```



4.بعد تمثيل الخلايا الثابتة وخلايا الحل كحقائق وهي حل صحيح للعبة سنبدأ بتعريف الأجرائيات التي تتأكد من صحة الحل بحيث تكون قواعد اللعبة محققة:

في البداية قمنا بتعريف اجرائية_ cell_ تتمثل بثلاث معاملات هي رقم السطر ورقم العامود والرمز الموجود بالخلية الناتجة عن تقاطع السطر مع العامود حيث تفيد هذا الاجرائية في ارجاع الرمز الموجود بخلية معينة ضمن الرقعة الذي سوف يفيدنا في الاجرائيات اللاحقة

- طريقة تحقيق هذا الاجرائية

```
cell(X,Y,S):- solve_cell(X,Y,S);fixed_cell(X,Y,S).
```

ثم قمنا بتعريف اجرائية_ get_row_التي تتمثل بمعاملين رقم السطر ومتغير لوضع عناصر السطر فيه , هذا الاجرائية نقوم يأرجاع عناصر سطر محدد بشكل مرتب حيث هي دالة مضمنة لل_ cell_ التي تقوم باستدعاء خلايا السطر بشكل متسلسل .

- طريقة تحقيق هذا الاجرائية

```
get_row(RowNum,Row) :- cell(RowNum, 1, X1),cell(RowNum, 2,  X2),
cell(RowNum, 3, X3),cell(RowNum, 4, X4),cell(RowNum, 5, X5),
 cell(RowNum, 6, X6) ,Row = [X1,X2,X3,X4,X5,X6],!.
```

- اختبار الاجرائية

```
?- get_row(1,Row).
   Row = [x, o, x, o, x, o].
```

ثم قمنا بتعريف اجرائية _get_column_التي تتمثل بمعاملين رقم العامود ومتغير لوضع عناصر العامود فيه , هذا الاجرائية نقوم بأرجاع عناصر عامود محدد بشكل مرتب حيث هي دالة مضمنة لل cell التي تقوم باستدعاء خلايا العامود بشكل متسلسل

- طريقة تحقيق هذا الاجرائية

```
get_column(ColumnNum,Column) :- cell(1, ColumnNum, X1),cell(2, ColumnNum, X2),
cell(3, ColumnNum, X3),cell(4, ColumnNum, X4),cell(5, ColumnNum, X5),
cell(6, ColumnNum, X6),Column = [X1,X2,X3,X4,X5,X6],!.
```

- اختبار الاجرائية

```
?- get_column(1,Column).
   Column = [x, o, x, x, o, o].
```

بعدها قمنا بتعريف اجرائية _count_ التي تتمثل بثلاث معاملات سلسلة ومتغير المراد عده ومتغير نضع فيه النتيجة , هذا الاجرائية تقوم بأرجاع عدد رمز ما ضمن سلسة الذي يفيد في تحقيق احد قواعد اللعبة , حيث هي دالة عودية تقوم بستدعاء نفسها حتى تنتهي السلسة وفي كل مرة تجد هذا الرمز تقوم بزيادة العداد بمقدار واحد.

- طريقة تحقيق هذا الاجرائية

```
count([], _, 0).
count([H|T], X, C) :-count(T, X, C1),(X = H -> C is C1 + 1 ; C = C1).
```

- اختبار الاجرائية

```
?- count([x,o,x,o,o,x],x,R).
        R = 3.
```

ثم قمنا بتعريف اجرائية _agree_التي تتمثل بمعاملين هم سلسلتين , هذا الاجرائية نقومبفحص تطابق السلسلتين,الذي ذلك في اختبار عدم تطابق سلسلتين .

- طريقة تحقيق هذا الاجرائية

```
agree(S1,S2):- not(S1=S2).
```

- اختبار الاجرائية

```
?- agree([x,o,o,x,o,x],[o,o,x,o,x,x]).
true.
```

بعدها قمنا بتعريف اجرائية _all_cells_filled_ تأخذ معامل واحد, هذا الاجرائية تقوم بفحص ان جميع الخلايا تحتوي رمز , حيث هي دالة عودية تقوم بستدعاء نفسها حتى يتحقق شرط التوقف وايضا" هي مضمنة لكلا الدوال _get_column,get_row,check_listحيثcheck_list هي التي تقوم بفحص كل خلية اذ تحوي احد الرمزين (Y,X) .

- طريقة تحقيق هذا الاجرائية

```
all_cells_filled :- all_cells_filled_helper(1).
all_cells_filled_helper(7) :- !.
all_cells_filled_helper(Z) :-
 get_row(Z, Row),
 get_column(Z, Column),
 check_list(Row),
 check_list(Column),
 Z1 is Z + 1,
 all_cells_filled_helper(Z1).
check_list([]).
check_list([H|T]) :- (H == x ; H == o), check_list(T).
```

- اختبار الاجرائية

```
?- all_cells_filled .
        true
```

بعدها قمنا بتعريف اجرائية _no_tripple لاتأخذ معاملات, تقوم بالتحقق من عدم تكرار عنصر 3 مرات متتاليات, حيث هي دالة عودية تقوم بستدعاء نفسها حتى يتحقق شرط التوقف وايضا" هي مضمنة لكلا الدوال _get_column,get_row,check_tripple _حيث check_tripple هي التي تقوم بفحص عدم تكرار عنصر 3 مرات متتاليات .

- طريقة تحقيق هذا الاجرائية

```
no_tripple:- no_tripple_helper(1).
no_tripple_helper(7):-!.
no_tripple_helper(C):-
get_row(C,List),
check_tripple(List),
get_column(C,List1),
check_tripple(List1),
C1 is C+1,
no_tripple_helper(C1).
check_tripple(Xs):-
  append(_, [X, X, X | _], Xs),!,false.
check_tripple(_).
```

- اختبار الاجرائية

```
?- no_tripple.
        true
```

بعدها قمنا بتعريف اجرائية _sympol_count_correct_ لاتأخذ معاملات, تقوم بالتأكد بان عدد Xهو نفسه عدد O في السلسلة, حيث هي دالة عودية تقوم بستدعاء نفسها حتى يتحقق شرط التوقف وايضا" هي مضمنة لكلا الدوال _get_column,count,get_row,length_ حيث length تقوم بارجاع طول السلسلة .

- طريقة تحقيق هذا الاجرائية

```
sympol_count_correct:- sympol_count_correct_helper(1).
sympol_count_correct_helper(7):- !.
sympol_count_correct_helper(C) :-get_row(C,List),
  length(List, Length),
  HalfLength is Length / 2,
  count(List,x ,CountA),
  count(List,o,CountB),
  CountA =:= HalfLength,
  CountB =:= HalfLength,
  get_column(C,List1),count(List,x,CountC),
  count(List,o,CountD),
  CountC =:= HalfLength,
  CountD =:= HalfLength,C1 is
C+1,sympol_count_correct_helper(C1).
```

ثم قمنا بتعريف اجرائية _ no_repeat _ لاتأخذ معاملات, تقوم بالتحق من عدم تطابق سطر مع سطر او عامود مع عامود ولذلك لتحقيق احد شروط اللعبة , حيث هي دالة تقوم بستدعاء دلين _no_repeat_cols_ التي تقوم بستدعاء الدالة _ get_all_cols_التي تقوم بجلب كل الاعمدة ثم استدعاءالدالة_check_rows_or_cols_لتحقق من عدم تطابق عامود مع عامود ,الدالة الثانية هي _ no_repeat_rows _ التي تقوم بستدعاء الدالة _ get_all_cols_التي تقوم بجلب كل الاعمدة ثم استدعاء_ get_all_rows_ التي تقوم بجلب كل الاسطر ثم استدعاءالدالة _ check_rows_or_cols_لتحقق من عدم تطابق سطر مع سطر .

- طريقة تحقيق هذا الاجرائية

```
no_repeat:- no_repeat_cols,no_repeat_rows.

no_repeat_cols:- get_all_cols(Board),check_rows_or_cols(Board).

no_repeat_rows:- get_all_rows(Board),check_rows_or_cols(Board).

no_repeat_rows_and_cols:- get_all_rows(Board),check_rows_and_cols(Board).

check_rows_or_cols(Board):-

    \+ (select(Row1, Board, Rest), member(Row2, Rest), Row1 = Row2).
```

اخيرا بالقسم الاول من المشروع بعد انا عرفنا جميع اجرائيات التي تحقق اللعبة قمنا بتهيئة تابع الحل الذي يقوم بستدعاء جميع الاجرائيات معا لتحقق من صحة الحل الذي فرضنه انه صحيح من اجل قواعد اللعبة

- طريقة تحقيق هذا الاجرائية

```
solved:- all_cells_filled,
 no_tripple,
 sympol_count_correct,
 no_repeat.
```

- اختبار الاجرائية

```
?- solved.
    true
```

في النهاية نستنتج بان الاجرائيات التي تحقق قواعد اللعبة تحققت اذ الحل الذي فرضناه صحيح.

في كل ماسبق كنا فارضين حل صحيح وعلى اساسو اختبرنا الأجرائيات التي تحقق قواعد اللعبة ,اما في حال اردنا اللاعب هو من يقوم بالأدخال لخلايا الحل سنشرح هذا الحالة هنا.

- في البداية عرفنا نفس الرقعة ب (36)حالة ,حيث يوجد(9)حالات منها ثابتة و(27) حالة منها متغيرة اما ان تحتوي ع(O)او(X)حسب رغبت اللاعب وفهمه للاعبه للوصول الى حل صحيح.
- بعد ذلك قمنا باستخدام قاعدة البيانات الديناميكية في بيئة Prolog لتخزين خلايا الرقعة بالاضافة للاجرائيات التي عرفناها في الجزء الاول من اللعبة ,لكن ضفنا بعض الاجرائيات سنشرح الية كتابتها وطريقةعملها
- اول دالة جديدة هي _solved_ حيث تقوم بالتحقق من الحل في جزء من اللعبة وترجع للاعب رسالة تبين فيها نوع الخطأ الذي تجاوز فيه قواعد اللعبة
- طريقة تحقيق هذا الاجرائية والرسالة التي ترجعها

```
solved :-
  (
  \+ all_cells_filled % false
  -> write('All slots must be filled'), nl, fail
  ;
  (
   \+ no_tripple % false
   -> write('It is forbidden to repeat x or o 3 times in a row or column'), nl, fail
    ;
  (
   \+ sympol_count_correct % false
   -> write('Make sure that the number of x and o is the same in all rows and columns'), nl, fail
    ;
   (
    \+ no_repeat % false
     -> write('Make sure that no lines or columns are the same'), nl, fail
     ; % true
     write('Well done, you solved the puzzle successfully'), nl, true
    )
   )
  )
  ).
```

- دالة تقوم بطباعة الرقعة
- طريقة تحقيق هذا الاجرائية

```
print_board :- get_all_rows(Board), print_board_helper(Board).

print_board_helper([]).
print_board_helper([H|T]) :- write(H), nl, print_board_helper(T).
```

- دالة تمكن اللاعب من حذف جميع الادخالات والعودة للبداية من جديد clear
- طريقة تحقيق هذا الاجرائية

```
clear:- retractall(solve_cell(X,Y,Z)),
asserta(solve_cell(1, 1,'_')),
asserta(solve_cell(1, 2,'_')),
asserta(solve_cell(1, 4,'_')),
asserta(solve_cell(1, 5,'_')),
asserta(solve_cell(1, 6,'_')),
asserta(solve_cell(2, 1,'_')),
asserta(solve_cell(2, 2,'_')),
asserta(solve_cell(2, 4,'_')),
asserta(solve_cell(2, 5,'_')),
asserta(solve_cell(2, 6,'_')),
asserta(solve_cell(3, 2,'_')),
asserta(solve_cell(3, 3,'_')),
asserta(solve_cell(3, 4,'_')),
asserta(solve_cell(3, 5,'_')),
asserta(solve_cell(4, 1,'_')),
asserta(solve_cell(4, 2,'_')),
asserta(solve_cell(4, 4,'_')),
asserta(solve_cell(4, 5,'_')),
asserta(solve_cell(4, 6,'_')),
asserta(solve_cell(5, 1,'_')),
asserta(solve_cell(5, 3,'_')),
asserta(solve_cell(5, 4,'_')),
asserta(solve_cell(5, 5,'_')),
asserta(solve_cell(6, 2,'_')),
asserta(solve_cell(6, 3,'_')),
asserta(solve_cell(6, 4,'_')),
asserta(solve_cell(6, 6,'_')).
```

- اخيرا دالة اللعب التي تقوم بالاتصال بقاعدة البيانات و الوصول لاجرئيات اللعبة
- طريقة تحقيق هذا الاجرائية

```
play:- write('Enter row please: '),read(R),write('Enter column please:
'),read(C),check(R,C),write('Enter x or o:
'),read(CH),retractall(solve_cell(R,C,_)),asserta(solve_cell(R,C,CH)),print_board.
```

- **في القسم الثاني من المشروع تم تنجيز ا الاستراتيجيات الاولى ,يلي بخبر فيها الحاسوب بالمرور على الصفوف والاعمدة في حال وجد ذات الرمز متتالي مرتين في سطر او عامود,يقوم بوضع الرمز المعاكس على كلا الجانبين لتجنب تكرار الثلاثيات.**

- **طريقة تحقيق هذا الاجرائية**

```
process_array(List, Result) :-
    process_start(List, StartProcessed),
    process_middle(StartProcessed, MiddleProcessed),
    process_end(MiddleProcessed, Result).

% المعالجة في بداية المصفوفة
process_start([x, x,_| Rest], [x, x, o | Rest]) :- !.
process_start([o, o,_ | Rest], [o, o, x | Rest]) :- !.
process_start(List, List).

% المعالجة في منتصف المصفوفة
process_middle([_,x,x_,_,_], [o,x,x,o,_,_]) :- !.
process_middle([_,o,o_,_,_], [x,o,o,x,_,_]) :- !.
process_middle([_,_,x,x,_,_], [_,o,x,x,o,_]) :- !.
process_middle([_,_,o,o,_,_], [_,x,o,o,x,_]) :- !.
process_middle([_,_,_,x,x,_], [_,_,o,x,x,o]) :- !.
process_middle([_,_,_,o,o,_], [_,_,x,o,o,x]) :- !.
process_middle(List, List).

% المعالجة في نهاية المصفوفة
process_end([_,_,_,_, x, x], [_,_,_,o, x, x]) :- !.
process_end([_,_,_,_, o, o], [_,_,_,x, o, o]) :- !.
process_end(List, List).
```

- **طريقة تحقيق هذا الاجرائية**

```
?- process_start(List, List).
List = [x, x, o|_].

?- process_middle(List, List).
List = [_, o, x, x, o, _].

?- process_end(List, List).
List = [_, _, _, o, x, x].
```

- و تم تنجيز ا الاستراتيجيات الثانية ,يلي بخبر فيها الحاسوب بالمرور على الصفوف والاعمدة في حال وجد ذات الرمز متتالي مرتين في في ا سطر او عامود ويوجد فراغ بين التكرار الاول والتكرار الثاني ,يقوم بوضع الرمز المعاكس في هذا الفراغ .

- طريقة تحقيق هذا الاجرائية

```
opposite(x, o).
opposite(o, x).
replace_middle([], []).
replace_middle([X], [X]).
replace_middle([X, Y], [X, Y]).

replace_middle([X, _, Y | T], [X, New, Y | NewT]) :-
    opposite(X, OppositeX),
    opposite(Y, OppositeY),
   New = OppositeX,
     New = OppositeY,
     replace_middle(T, NewT).
```

- طريقة تحقيق هذا الاجرائية

```
[debug]  ?- get_row(1,Row),replace_middle(Row,Result).
Row = [x, '_', x, o, '_', o],
Result = [x, o, x, o, x, o].
```

- و تم تنجيز ا الاستراتيجيات الثالثة,يلي بخبر فيها الحاسوب بالمرور على الصفوف والاعمدة ويقوم بفحص تكرارات الرموز وعدها ووضع الرمز المناسب حسب العدد بشرط تجنب الثلاثيات .

- طريقة تحقيق هذا الاجرائية

```
avoidingtriples([-|T], Res) :-
% Find the first row that has an empty cell in the first col
Res = [x|T],
% Check if the puzzle satisfies the conditions
(no_tripple_helper(Res), sympol_count_correct_helper(Res) -> true) ;
% If the puzzle doesn't satisfy the conditions, try the other symbol
Res = [o|T],
% Check if the puzzle satisfies the conditions
(no_tripple_helper(Res), sympol_count_correct_helper(Res) -> true).
```

- طريقة تحقيق هذا الاجرائية

```
[debug]  ?- avoidingtriples([-,x,-,-,o,x,x,o],R).
R = [o, x, -, -, o, x, x, o].
```

STUDENTS NAMES

Amen AL Nuserat
Anas AL Hiemad
Aya Mousa
wael drdr
baraa dali hassan

Faculty of Informatics Engineering, Damascus University