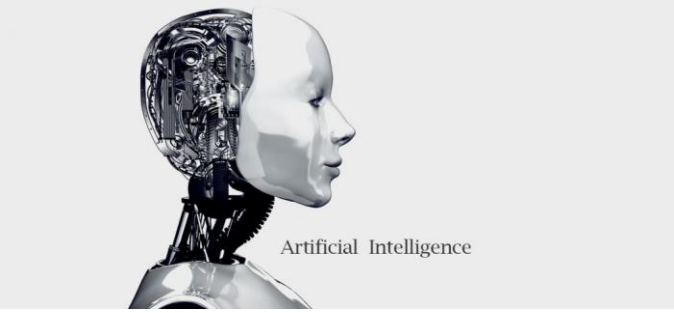

Artificial Intelligence Project

إشراف المهندس: فراس سلمان

إعداد الطلاب: محمد عيد التل – لبنى ديوب – عبد الرحمن عيطة – عبد الرحمن الحوراني – حنين مسعود – زكريا الشيخ

الاسنادية initTop تقوم بتهيئة جميع الحقائق top(X,Y) حيث X يمثل رقم العمود و Y يمثل عدد القطع الموجودة في العمود

```
initTop(0,Y):-!.
initTop(X,Y):- assert(top(X,Y)),X1 is X-1, initTop(X1,Y).
```

الاسنادية removeTop تقوم بحذف حقيقة top(X,Y)

الاسنادية removeAllTop تقوم بحذف جميع الحقائق top(X,Y)

```
removeTop(X,Y):- retract(top(X, Y)).
removeAllTop():- top(X, Y),once(removeTop(X,Y)),fail.
```

الاسنادية addSize تقوم بتهيئة حقيقة size(X,Y)

الاسنادية removeSize تقوم بحذف جميع الحقائق size(X,Y)

```
addSize(X, Y):- assert(size(X,Y)).

removeSize():- size(X, Y), once(retract(size(X, Y))), fail.
```

الاسنادية add(X,Y) تقوم بإضافة حقيقة size(X,Y) تعبر عن حجم الرقعة وتقوم بتهيئة جميع عواميد الرقعة بالرقم صفر حيث تكون الأعمدة جميعها في بداية اللعبة فارغة

```
add(X, Y):- addSize(X,Y), initTop(X,0).
```

الاسنادية remove() تقوم بحذف جميع الحقائق size(X,Y) , top(X,Y) , piece(X,Y,Z)

```
remove():- addSize(0,0),assert(top(0,0)),assert(piece(0,0,0)),
           not(removeSize()),not(remove_All_piece()),not(removeAllTop()).
```

الاسنادية canWeAdd(X) تقوم بالتأكد من أنه يمكن إضافة قطعة في عمود ما

```
canWeAdd(X):- size(_, Y), top(X, N), N<Y.
```

الاسنادية removeLastTopAndAddNewOne(X,Y) تقوم بحذف آخر حقيقة top(X,Y) وتهيئة حقيقة جديدة مع زيادة عدد القطع في العمود بمقدار 1

```
removeLastTopAndAddNewOne(X,Y):- top(X,L),removeTop(X,L),Y is L+1,assert(top(X,Y)).
```

الاسنادية add_piece(X,Z) يقوم باضافة قطعة حيث X يمثل رقم العمود الذي نريد وضع القطعة فيه و Z يمثل لون القطعة

شرح بسيط: يقوم بالتحقق من امكانية اضافة القطعة عن طريق الاسنادية canWeAdd , يقوم بإضافة قطعة للعمود عن طريق الاسنادية removeLastTopAndAddNewOne ثم يقوم بإضافة حقيقة piece(X,Y,Z)

```
add_piece(X,Z):- canWeAdd(X),removeLastTopAndAddNewOne(X,Y),assert(piece(X,Y,Z)).
```

الاسنادية remove_piece(X,Y,Z) تقوم بحذف حقيقة piece(X,Y,Z) أي تقوم بحذف قطعة

الاسنادية remove_All_piece تقوم بحذف جميع الحقائق piece(X,Y,Z)

```
remove_piece(X,Y,Z):- top(X,T),Y=T,removeTop(X,T),T1 is T - 1,assert(top(X,T1)),retract(piece(X,Y,Z)).
remove_All_piece():- piece(X,Y,Z),once(retract(piece(X,Y,Z))),fail.
```

الاسنادية writeGame() تقوم بطباعة الرقعة

الاسنادية writee(X, Y) تقوم بطباعة سطر من الرقعة

```
writeLine(Y):- size(N,_),Y =< N, Y1 is Y + 1, write('======'), writeLine(Y1).

writeGame():- size(_,N), not(writeee(N+1)),writeLine(0).

writeee(X):- X1 is X - 1,X1 >= 1, write('|'), not(writee(X1, 1)),write('|\n'),writeee(X1).

writee(X, Y):- size(N,_),Y=<N,piece(Y,X,Z),write(Z),write('\t'),Y1 is Y + 1,writee(X, Y1),!;
               size(N,_),Y=<N,not(piece(Y,X,Z)),write('not\t'),Y1 is Y + 1,writee(X, Y1),!.
```

الاسنادية choseColor(Player, Computer) تقوم بتحديد لون المستخدم ولون الكمبيوتر

```
choseColor(Player, Computer):- write('\nPlayer Color = '),read(Player),
                               write('\nComputer Color = '),read(Computer).
```

الاسنادية win(Col, Player) حيث الـ Col يمثل العمود والـ Player يمثل القطعة

نعطيه رقم العمود Col ونحصل على رقم السطر عن طريق الحقيقة top وفي حال كانت القطعة التي نريد اختبارها رابحة يعطينا false ويذهب للجزء الثاني من الاسنادية وهو الطباعة

```
win(Col, Player):- top(Col, Row),not(win(Col, Row, Player,4)),!;
                   write('win = '),write(Player),write('\n'),not(fail).
```

الاسنادية getFirst([X|T],X) تقوم بأخذ أول عنصر من ليست

الاسنادية minTop(X) تقوم بأخذ أصغر عدد قطع موجودة في عمود ما من الرقعة الهدف من ذلك ليلعب الكمبيوتر في العمود الذي يوجد فيه أقل عدد قطع

```
getFirst([X|T],X).
minTop(X):- findall(Here,top(A,Here),T),sort(T,T1),getFirst(T1,X).
```

الاسنادية checkLeft(X,Y,Z,A) تقوم بعد القطع التي على يسار القطعة المحددة وتعطي false في حال كان اللون مختلف وتتوقف الاسنادية في حال وصلنا للعمود 0

الاسنادية checkRight(X,Y,Z,A) نفس الاسنادية السابقة لكنها بالاتجاه اليميني وتتوقف الاسنادية في حال وصلنا لآخر عمود في الرقعة

الاسنادية checkHorizontal(X,Y,Z,A) تستدعي الاسناديتين السابقتين وتضع الناتج في متحول

```
checkLeft(0,Y,0,A):-!.
checkLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,checkLeft(X1,Y,Z1,A),Z is Z1+1,!;Z is 0.

checkRight(K,Y,1,A):-size(M,N),K=M,!.
checkRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,checkRight(X1,Y,Z1,A),Z is Z1+1,!;Z is 0.

checkHorizontal(X,Y,Z,A):- checkLeft(X,Y,Z1,A),checkRight(X,Y,Z2,A),Z is Z1+Z2-1.
```

الاسنادية checkDown(X,Y,Z,A) تقوم بعد القطع تحت القطعة المحددة وتعطي false في حال كان اللون مختلف وتتوقف الاسنادية في حال وصلنا للسطر 0

الاسنادية checkVertical(X,Y,Z,A) هي فقط استدعاء للاسنادية السابقة

```
checkDown(X,0,0,A):-!.
checkDown(X,Y,Z,A):- piece(X,Y,B), A=B,Y1 is Y-1,checkDown(X,Y1,Z1,A),Z is Z1+1,!;Z is 0.

checkVertical(X,Y,Z,A):- checkDown(X,Y,Z1,A),Z is Z1.
```

الاسنادية checkUpLeft(X,Y,Z,A) تقوم بعد القطع من القطعة المحددة لتصل إلى الشمال الغربي (أعلى اليسار) وفي كل مرة تنتقل لسطر أعلى وترجع عمود إلى الخلف

الاسنادية checkDownRight(X,Y,Z,A) تقوم بعد القطع من القطعة المحددة لتصل إلى الجنوب الشرقي (أسفل اليمين) وفي كل مرة تنتقل لسطر أسفل وعمود أعلى

الاسنادية checkScale1(X,Y,Z,A) تستدعي الاسناديتين السابقتين وتضع الناتج في متحول

```
checkUpLeft(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkUpLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,Y1 is Y-1,checkUpLeft(X1,Y1,Z1,A),Z is Z1+1,!;Z is 0.

checkDownRight(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkDownRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,Y1 is Y+1,checkDownRight(X1,Y1,Z1,A),Z is Z1+1,!;
                          Z is 0.

checkScale1(X,Y,Z,A):- checkUpLeft(X,Y,Z1,A),checkDownRight(X,Y,Z2,A),Z is Z1+Z2-1.
```

الاسنادية checkDownLeft(X,Y,Z,A) تقوم بعد القطع من القطعة المحددة لتصل إلى الشمال الشرقي (أعلى اليمين)

الاسنادية checkUpRight(X,Y,Z,A) تقوم بعد القطع من القطعة المحددة لتصل إلى الجنوب الغربي (أسفل اليسار)

الاسنادية checkScale2(X,Y,Z,A) تستدعي الاسناديتين السابقتين وتضع الناتج في متحول

```
checkDownLeft(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkDownLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,Y1 is Y+1,checkDownLeft(X1,Y1,Z1,A),Z is Z1+1,!;
                         Z is 0.

checkUpRight(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkUpRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,Y1 is Y-1,checkUpRight(X1,Y1,Z1,A),Z is Z1+1,!;Z is 0.

checkScale2(X,Y,Z,A):- checkDownLeft(X,Y,Z1,A),checkUpRight(X,Y,Z2,A),Z is Z1+Z2-1.
```

الاسنادية win(X,Y,A,Num) حيث X هو العمود والـ Y هو السطر والـ A هو لون القطعة والـ Num عدد مرات تكرار القطعة

شرح بسيط: تقوم بعد القطع التي من نفس اللون شاقولياً أوعمودياً أ وقطرياً بالاتجاهين من خلال الاسناديات التي عرفناها سابقاً وتكون اللعبة رابحة في حال كان الرقم أكبر أو يساوي 4

```
win(X,Y,A,Num):- checkHorizontal(X,Y,Z,A),Z >= Num,!;
                 checkVertical(X,Y,Z,A),Z >= Num,!;
                 checkScale1(X,Y,Z,A),Z >= Num,!;
                 checkScale2(X,Y,Z,A),Z >= Num,!.
```

الاسنادية numberOfPiece(N) ترد عدد القطع الموجودة في الرقعة

الاسنادية winn(Col, Player,Num) نفس الاسنادية السابقة لكنها تختلف بالبارمتر الأخير الذي يحدد بكم حجر يمكنك الفوز

الاسنادية computerTurn(Computer) سنحصل على أقل عدد قطع موجودة في عمود ما والتي تمثل رقم السطر الذي وصلنا اليه , بعدها سنأخذ رقم العمود الذي يحتوي هذه القطع ونضيف القطعة بهذا المكان ونختبر الفوز

الاسنادية computerTurn(Computer,N,T) :

1. اذا كان العمود ممتلئ سيرد false
2. اذا أضفنا قطعة وكانت رابحة سيتوقف ويرجع false
3. اذا أضفنا قطعة وكانت خاسرة سيتراجع عن القطعة التي أضافها ويستدعي نفسه من أجل العمود التالي

الاسنادية c(Computer):

1. اذا وجدنا امكانية اضافة حجر ليشكل 4 احجار متتالية نضيفه ونخرج
2. اذا وجدنا امكانية اضافة حجر ليشكل 3 احجار متتالية نضيفه ونخرج
3. اذا وجدنا امكانية اضافة حجر ليشكل 2 حجرين متتالية نضيفه ونخرج
4. وإلا نضيف الحجر بأي مكان في الرقعة

```
numberOfPiece(N):- findall(Here, piece(Other1, Other2, Here), X),length(X, N).


c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,4)),numberOfPiece(N1),N1 > N,!.
c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,3)),numberOfPiece(N1),N1 > N,!.
c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,2)),numberOfPiece(N1),N1 > N,!.
c(Computer):- computerTurn(Computer).


computerTurn(Computer,N,T):- top(N,X),add_piece(N, Computer),!,
                             not(winn(N, Computer,T)),X1 is X + 1,remove_piece(N,X1,Computer),N1 is N + 1,
                             computerTurn(Computer, N1,T),!;
                             top(N,X),not(add_piece(N, Computer)),N1 is N + 1, computerTurn(Computer, N1,T).


computerTurn(Computer):- minTop(X),once(top(N,X)),add_piece(N, Computer),win(N, Computer).


winn(Col, Player,Num):- top(Col, Row),not(win(Col, Row, Player,Num)),!;
                        write('win = '),write(Player),write('\n'),not(fail).
```

الاسنادية start(X,Y)

1. نقوم بتهيئة اللعبة عن طريق الاسناديتين remove(),add(X,Y)
2. نقوم بتحديد الألوان عن طريق الاسنادية choseColor(Player, Computer)
3. Repeat ستقوم بتكرار التالي:
   - نأخذ رقم العمود الذي يريد اللاعب اضافة القطعة فيه
   - اضافة قطعة اللاعب في العمود المحدد
   - نختبر فوز اللاعب
   - يلعب الكمبيوتر

```
getOut(z).

start(X,Y):-  remove(),add(X,Y),choseColor(Player, Computer),repeat,
        not(writeGame()),
        write('\nColumns = '),read(Col),add_piece(Col, Player),win(Col, Player),not(c(Computer)),
        (getOut(Col),getOut(Coll);fail).
```

صورة عن التنفيذ

```
?- start(5,4).

Player Color = p.

Computer Color = |: c.
|not    not     not     not     not     |
|not    not     not     not     not     |
|not    not     not     not     not     |
|not    not     not     not     not     |
====================================
Columns = |: 1.
|not    not     not     not     not     |
|not    not     not     not     not     |
|c      not     not     not     not     |
|p      not     not     not     not     |
====================================
Columns = |: 2.
|not    not     not     not     not     |
|c      not     not     not     not     |
|c      not     not     not     not     |
|p      p       not     not     not     |
====================================
Columns = |: 3.
|c      not     not     not     not     |
|c      not     not     not     not     |
|c      not     not     not     not     |
|p      p       p       not     not     |
====================================
Columns = |: 4.
win = p
|c      not     not     not     not     |
|c      not     not     not     not     |
|c      c       not     not     not     |
|p      p       p       p       not     |
====================================
```

Prolog Code

```
initTop(0,Y):-!.
initTop(X,Y):- assert(top(X,Y)),X1 is X-1, initTop(X1,Y).


removeTop(X,Y):- retract(top(X, Y)).
removeAllTop():- top(X, Y),once(removeTop(X,Y)),fail.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
addSize(X, Y):- assert(size(X,Y)).
removeSize():- size(X, Y), once(retract(size(X, Y))), fail.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
add(X, Y):- addSize(X,Y), initTop(X,0).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
remove():- addSize(0,0),assert(top(0,0)),assert(piece(0,0,0)),
        not(removeSize()),not(remove_All_piece()),not(removeAllTop()).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
canWeAdd(X):- size(_, Y), top(X, N), N<Y.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
removeLastTopAndAddNewOne(X,Y):- top(X,L),removeTop(X,L),Y is L+1,assert(top(X,Y)).


add_piece(X,Z):- canWeAdd(X),removeLastTopAndAddNewOne(X,Y),assert(piece(X,Y,Z)).


remove_piece(X,Y,Z):- top(X,T),Y=T,removeTop(X,T),T1 is T - 1,assert(top(X,T1)),retract(piece(X,Y,Z)).
remove_All_piece():- piece(X,Y,Z),once(retract(piece(X,Y,Z))),fail.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
writeLine(Y):- size(N,_),Y =< N, Y1 is Y + 1, write('======'), writeLine(Y1).


writeGame():- size(_,N), not(writeee(N+1)),writeLine(0).


writeee(X):- X1 is X - 1,X1 >= 1, write('|'), not(writee(X1, 1)),write('|\n'),writeee(X1).
```

```
writee(X, Y):- size(N,_),Y=<N,piece(Y,X,Z),write(Z),write('\t'),Y1 is Y + 1,writee(X, Y1),!;
          size(N,_),Y=<N,not(piece(Y,X,Z)),write('not\t'),Y1 is Y + 1,writee(X, Y1),!.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
choseColor(Player, Computer):- write('\nPlayer Color = '),read(Player),
                    write('\nComputer Color = '),read(Computer).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
getFirst([X|T],X).
minTop(X):- findall(Here,top(A,Here),T),sort(T,T1),getFirst(T1,X).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
numberOfPiece(N):- findall(Here, piece(Other1, Other2, Here), X),length(X, N).
c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,4)),numberOfPiece(N1),N1 > N,!.
c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,3)),numberOfPiece(N1),N1 > N,!.
c(Computer):- numberOfPiece(N),not(computerTurn(Computer,1,2)),numberOfPiece(N1),N1 > N,!.
c(Computer):- computerTurn(Computer).



computerTurn(Computer,N,T):- top(N,X),add_piece(N, Computer),!,
                  not(winn(N, Computer,T)),X1 is X + 1,remove_piece(N,X1,Computer),N1 is N + 1,
                  computerTurn(Computer, N1,T),!;
                  top(N,X),not(add_piece(N, Computer)),N1 is N + 1, computerTurn(Computer, N1,T).



computerTurn(Computer):- minTop(X),once(top(N,X)),add_piece(N, Computer),win(N, Computer).



winn(Col, Player,Num):- top(Col, Row),not(win(Col, Row, Player,Num)),!;
              write('win = '),write(Player),write('\n'),not(fail).


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
win(Col, Player):- top(Col, Row),not(win(Col, Row, Player,4)),!;
            write('win = '),write(Player),write('\n'),not(fail).
```

```
getOut(z).


start(X,Y):-  remove(),add(X,Y),choseColor(Player, Computer),repeat,
      not(writeGame()),
      write('\nColumns = '),read(Col),add_piece(Col, Player),win(Col, Player),not(c(Computer)),
      (getOut(Col);fail).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
checkLeft(0,Y,0,A):-!.
checkLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,checkLeft(X1,Y,Z1,A),Z is Z1+1,!;Z is 0.


checkRight(K,Y,1,A):-size(M,N),K=M,!.
checkRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,checkRight(X1,Y,Z1,A),Z is Z1+1,!;Z is 0.


checkHorizontal(X,Y,Z,A):- checkLeft(X,Y,Z1,A),checkRight(X,Y,Z2,A),Z is Z1+Z2-1.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
checkDown(X,0,0,A):-!.
checkDown(X,Y,Z,A):- piece(X,Y,B), A=B,Y1 is Y-1,checkDown(X,Y1,Z1,A),Z is Z1+1,!;Z is 0.


checkVertical(X,Y,Z,A):- checkDown(X,Y,Z1,A),Z is Z1.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%


checkUpLeft(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkUpLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,Y1 is Y-1,checkUpLeft(X1,Y1,Z1,A),Z is Z1+1,!;Z is 0.


checkDownRight(X,Y,1,A):- size(M,N),X=M,Y=N,!.
checkDownRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,Y1 is Y+1,checkDownRight(X1,Y1,Z1,A),Z is Z1+1,!;
            Z is 0.


checkScale1(X,Y,Z,A):- checkUpLeft(X,Y,Z1,A),checkDownRight(X,Y,Z2,A),Z is Z1+Z2-1.
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

checkDownLeft(X,Y,1,A):- size(M,N),X=M,Y=N,!.

checkDownLeft(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X+1,Y1 is Y+1,checkDownLeft(X1,Y1,Z1,A),Z is Z1+1,!;

                    Z is 0.


checkUpRight(X,Y,1,A):- size(M,N),X=M,Y=N,!.

checkUpRight(X,Y,Z,A):- piece(X,Y,B),A=B,X1 is X-1,Y1 is Y-1,checkUpRight(X1,Y1,Z1,A),Z is Z1+1,!;Z is 0.


checkScale2(X,Y,Z,A):- checkDownLeft(X,Y,Z1,A),checkUpRight(X,Y,Z2,A),Z is Z1+Z2-1.


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

win(X,Y,A,Num):- checkHorizontal(X,Y,Z,A),Z >= Num,!;

          checkVertical(X,Y,Z,A),Z >= Num,!;

          checkScale1(X,Y,Z,A),Z >= Num,!;

          checkScale2(X,Y,Z,A),Z >= Num,!.
```