Implement Tic Tac Toe Game

```
My move
['', '', '']
['', 'O', '']
['', '', '']
Player 2 : 0 0
['X', '', '']
['', 'O', '']
['', '', '']
My move
['X', '', '']
['', 'O', '']
['O', '', '']
Player 2 : 0 2
['X', '', 'X']
['', 'O', '']
['O', '', '']
My move
No probable win
Row might win 0
['X', 'O', 'X']
['', 'O', '']
['O', '', '']
Player 2 : 2 1
['X', 'O', 'X']
['', 'O', '']
['O', 'X', '']
My move
No probable win
No probable win
['X', 'O', 'X']
['O', 'O', '']
['O', 'X', '']
Player 2 : 1 2
['X', 'O', 'X']
['O', 'O', 'X']
['O', 'X', '']
My move
No probable win
Col might win 2
['X', 'O', 'X']
['O', 'O', 'X']
['O', 'X', 'O']
Its a tie
```

Solve 8-Puzzle Problem

```
Sequence: 123046758
Number of moves - 3
Number of traversed nodes - 8
```

Implement Vacuum Cleaner Agent

```
State: {'A': 'Dirty', 'B': 'Dirty'}
Vacuum Cleaner Location: A
Action: Suck
State: {'A': 'Clean', 'B': 'Dirty'}
Vacuum Cleaner Location: A
Action: Right
State: {'A': 'Clean', 'B': 'Dirty'}
Vacuum Cleaner Location: B
Action: Suck
State: {'A': 'Clean', 'B': 'Clean'}
Vacuum Cleaner Location: B
Action: Left
State: {'A': 'Clean', 'B': 'Clean'}
Vacuum Cleaner Location: A
Action: Right
State: {'A': 'Clean', 'B': 'Clean'}
Vacuum Cleaner Location: B
Action: Left
State: {'A': 'Clean', 'B': 'Clean'}
```

Implement A* Search

```
123456780
123456708
123406758
123046758
```

Iterative Deepening to Solve 8 Puzzle Problem

```
Initial state
[1, 5, 2]
[4, 8, 0]
[7, 6, 3]
Depth:  1
Depth:  2
Depth:  3
Depth:  4
Depth:  5
Depth:  6
Depth:  7
Goal reached
```

Create a Knowledge Base with Propositional Logic to Show that a given Query Entails the Knowledge Base

```
Enter rule: pvq^(p^q)
Enter query: p^q^(pvq)
Truth table:
P        Q       Knowledge Base  Query
False    False   False   False
False    True    False   False
True     False   True    False
True     True    True    True
Knowledge base does not entail the query
```

Convert given Propositional Logic to CNF

```
a=>(bvc)
(bv~a)v(cv~a)
```

Implement Unification in First Order Logic

```
Original Expressions:
hate(f(y),z)
hate(M,f(y))
Substitutions:
[('M', 'f(y)'), ('M', 'z')]
```

Create a Knowledge Base consisting of First Order Logic Statements and Prove the Given Query using Forward Reasoning

```
New Clause: (American(x) & Weapon(y) & Sells(x, y, z) & Hostile(z)) ==> Criminal(x)
New Clause: Enemy(Nono, America)
New Clause: Owns(Nono, M1)
New Clause: Missile(M1)
New Clause: (Missile(x) & Owns(Nono, x)) ==> Sells(West, x, Nono)
New Clause: American(West)
New Clause: Missile(x) ==> Weapon(x)
New Clause:

0. Exit.
1. Tell Knowledge Base.
2. Ask Knowledge Base.
1

Tell: Enemy(Coco, America)
Tell: Enemy(Jojo, America)
Tell: Enemy(x, America) ==> Hostile(x)
Tell:

0. Exit.
1. Tell Knowledge Base.
2. Ask Knowledge Base.
2

Ask: Hostile(x)
[{x: Nono}, {x: Jojo}, {x: Coco}]
Ask: Criminal(x)
[{x: West}]
Ask:

0. Exit.
1. Tell Knowledge Base.
2. Ask Knowledge Base.
0
```

Demonstrate Decision Tree Learning

```
{'Outlook': {'overcast': 'yes', 'sunny': {'Humidity': {'high': 'no', 'normal': 'yes'}}, 'rainy': {'Windy': {'false': 'yes', 'true': 'no'}}}}
Query: rainy, mild, normal, false
yes
```