**Deterministic** – In computer science, a deterministic algorithm will always pass through the same sequence of “choices” in the same order, and with a given input will always produce the same output.

**The set of P problems** can best be described as the class of problems whose solutions are considered efficiently solvable in fixed Polynomial Time.

Examples: Primality testing

Testing for 3NF

Depth First Search

Circuit Value Problem

GCD computations

**The set of NP problems** are not easily solved, but can be easily *verified* as correct. The collection of problems with easily validated solutions is known as Nondeterministic Polynomial-Time.

Examples: Decomposition into 3NF

Graph Isomorphism Problem

Hamiltonian Circuits (Knights Tour)

3-Colors

Traveling Salesman

Games (Karuko, Sudoku, Tetris, etc)

KNIGHT’S TOUR

Let N be the size of the board (NxN). Let M = N2-1 (The number of valid moves the knight must make to make a complete tour. Let **visited** be a two dimensional NxN array of Boolean values, initialized to false and a location (x,y) in the array is set to true when the knight visits that spot. Let m be in [0, M] and represents the move number. Upon completion of the tour, m=M.

if( (x < 0) OR (x ≥ N) OR (y < 0) OR (y ≥ N) )

return false //A coordinate is off the board

if( visited[x, y] = true )

return false //Can’t move here; it has already been visited

if( m = M )

//This is a valid move and the knight has now made M moves; we have a solution

print “A solution has been found” print “ x, y ” //This starts printing the solution

set visited[x, y] = true

return true

else

//This is a valid move, but a tour has not been completed. Try all the moves that can be made from this location recursively.

let result be a Boolean variable //MUST be local set

result = false

set result = result OR Move( x+2, y+1, m+1)

set result = result OR Move( x+2, y-1, m+1)

set result = result OR Move( x-2, y+1, m+1)

set result = result OR Move( x-2, y-1, m+1)

set result = result OR Move( x+1, y+2, m+1)

set result = result OR Move( x+1, y-2, m+1)

set result = result OR Move( x-1, y+2, m+1)

set result = result OR Move( x-1, y-2, m+1)

if( result = true )

//One of the 8 moves above led to a completed tour. So, this position is part of a successful tour.

print “ x, y ”

return true

else

//None of the moves from this position led to a successful tour. Now we must backtrack and try a different path

set visited[x, y] = false //Un-visit this location

return false