**N-Queens Problem, PseudoCode**

*Samantha Husack*

***Recursive Method: PseudoCode***

Import Time Library

Open while loop: Continue Using Program

Define Safe\_Queen using, queens[] and column

for all values in vector queens

if current value is equal to an existing value or the column value minus current value is equal to the diagonal elements,

return false

return true if safe

Define solution using queens[] and column

if columns == size of board

add to counter 1 solution

print solution

create time stamp for execution length

find execution length (start time-end time)

print execution time since program launch

return

for all values in range(board size)

queen[column]value = current value

if safe\_queen(queens[], current value)

continue to new recursion :-> queens[] and col+1

n = input from user for board size as integer

start counter at 0 solutions found

create array of size n for queens found in solutions

start time for execution time count

find solution using queens[empty] and col = 0

when all solutions found, ask for input from user to calculate another solution, y/n

if == “n”, change while loop to stop running program.

else, restart program execution at n=input from user.

***Iterative Method : PseudoCode***

Import Time Library

Open while loop: Continue Using Program

Create Object: Board

Define initiate, using object and board\_size

Start solution counter

Initiate board\_size as N

Define safe\_queen, using object, row and col

For each val in queens[],

If the first elements r equal to current row, or if second element is equal to current col or is on the diagonal of one of the elements:

Not safe

Otherwise, safe queen

Define print\_board, using object

Print line of text with Solution ##, using counter

For each row in the range of 0-n

Define element for places where there is no queen

If there is a queen there, place Q,

Otherwise, place element ^^

end\_time for finding current solution from start of program

Print : difference in time in seconds for program execution time

Augment counter by 1

Print empty line

Define solution, using object

Create array queens[]

Set col and row to 0

While there are still possible solutions

While the col val is less than n and not in danger

Increase col by 1

If col smaller than n

Add col val to queens[]

If row +1 is greater or equal to n

Print board

Pop last queen

Col val is now N

Else

Increase row by 1

Set col to 0

If col is greater or equal to n

If row is equal to 0

Return, all combinations were tried

Pop the last queen and get next solution, increase col val by 1 to start at next position

Decrease row value by 1, go back a row

Define board\_size as input by user

Create new object Board, using board\_size

start\_time = for 1st point in program execution

Find solution of object

If next input is anything but “n”, restart program

If input is n : stop program execution.