AIProject
Harriet Willmott
Griffin Ledingham
Mark Lessard
=========

* Sudoku Solver*
works for 9x9 sudoku puzzles input in a variety of formats

The input data is represented in a file, in either 9 rows of numbers/symbols or one single long row
In the 9 row example, each row corresponds with a 9 column row in the sudoku puzzle.
If the puzzle is input in one single line, each 9 digits corresponds to a single row.

encodeMinimal() and encodeExtended() functions in Sudoku.py allow the program to encode the appropriate CNF's for each different case (each number appears at most once in each column, etc...)

findNextUnitClause(clauseList) in DPLL.py iterates through a provided input list of clauses. If the function finds a clause with length 1, return the clause's literal.

unitPropagateList(clauseList, partialAssignment, literalList) in DPLL.py iterates repeatedly through the clause list, checking for unit clauses, and on succesfully finding one, removes it negated counterpart from clauseList.

pureLiteral(element, clauseList) in DPLL.py checks the clauselist for all appearances of element (element being a pure literal). If element is found in a clause, the program removes that entire clause from clauseList.

runBacktracking(clauseList, partialAssignment, literalList) in DPLL.py is a recursive function which iterates through the possible combinations of literal assignments. Recursively calling the function for both positive, and negative assignments, the function iterates the combinations in a tree format, calling runBacktracing on positive and negative assignments, comparable to calling a function on a node's left and right children.