(setf gameBoard '((\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)(\_ \_ \_ \_ \_ \_ \_ \_)))

(setf columnPieces '(0 0 0 0 0 0 0 0))

(setq column 0)

(defun show()

(do ((i 7 (- i 1)))((= i -1) 'done)(format t "~A~%" (nth i gameBoard)))

(format t " 1 2 3 4 5 6 7 8~%"))

(defun get-choice()

(loop do

(format t "~%Column to play (0 to exit) ")

(setq column (- (read) 1))

(if (= column -1)

(return-from get-choice column))

(if (or (> -1 column)(< column 8))

(if (< (nth column columnPieces) 8)

(return-from get-choice column)

(princ "Column is full"))

(princ "Invalid column"))))

(defun get-computer-move()

(setf moveScores '(0 0 0 0 0 0 0 0))

(do ((i 0 (+ i 1)))((= i 8) 'done)

(if (< (nth i columnPieces) 8)

(progn

(setq score (check-in-a-row i (nth i columnPieces) "O"))

(setq score (+ score (check-in-a-row i (nth i columnPieces) "X"))))

(setq score -2))

(setf (nth i moveScores) score))

(setq bestScore (get-best-score moveScores))

(if (> bestScore 9999)

(return-from get-computer-move (get-best-move moveScores)))

(check-player-moves)

(setq bestScore (get-best-score moveScores))

(if (> bestScore 999)

(return-from get-computer-move (get-best-move moveScores)))

(check-next-moves)

(return-from get-computer-move (get-best-move moveScores)))

(defun get-best-move(scoreList)

(setq bestScore (get-best-score scoreList))

(setq bestMoves (count bestScore scoreList))

(if (> bestMoves 1)

(setq choice (random bestMoves))

(setq choice 0))

(do ((i 0 (+ i 1)))((= i 8) 'done)

(setq score (nth i scoreList))

(if (= score bestScore)

(progn

(if (= choice 0)

(return-from get-best-move i))

(setq choice (- choice 1))))))

(defun get-best-score (moveScores)

(setq best -2)

(dolist

(score moveScores best)

(if (> score best)

(setq best score))))

(defun check-player-moves ()

(do ((i 0 (+ i 1)))((= i 8) 'done)

(if (< (nth i columnPieces) 7)

(setq nextScore (check-in-a-row i (+ 1 (nth i columnPieces)) "X")))

(if (or (> nextScore 9999) (and (> nextScore 999)(< (nth i columnPieces) 1000)))

(setf (nth i moveScores) -1))))

(defun check-next-moves()

(do ((i 0 (+ i 1)))((= i 8) 'done)

(if (< -1 (nth i moveScores) 100)

(progn

(setf (nth i (nth (nth i columnPieces) gameBoard)) "O")

(incf (nth i columnPieces))

(setq theseScores (get-move-scores "O"))

(setq bestOfThese (get-best-score theseScores))

(if (and (> bestOfThese 1000) (> bestOfThese (nth i moveScores)))

(setf (nth i moveScores) bestOfThese))

(decf (nth i columnPieces))

(setf (nth i (nth (nth i columnPieces) gameBoard)) "\_")))))

(defun get-move-scores (piece)

(setf moveScoreList '(0 0 0 0 0 0 0 0))

(do ((i 0 (+ i 1)))((= i 8) moveScoreList)

(if (< (nth i columnPieces) 8)

(setq score (check-in-a-row i (nth i columnPieces) piece))

(setq score -2))

(setf (nth i moveScoreList) score)))

(defun get-piece (column row)

(if (or (or (< row 0)(> row 7)) (or (< column 0)(> column 7)))

(return-from get-piece "out")

(return-from get-piece (nth column (nth row gameBoard)))))

(defun get-number-in-a-row (column row columnOffsetOrig rowOffsetOrig checkPiece)

(setq continue 1)

(setq total 0)

(setq columnOffset columnOffsetOrig)

(setq rowOffset rowOffsetOrig)

(loop while (= continue 1) do

(setq thisPiece (get-piece (+ column columnOffset) (+ row rowOffset)))

(if (string/= checkPiece thisPiece)

(setq continue 0)

(progn

(setq total (+ total 1))

(setq columnOffset (+ columnOffset columnOffsetOrig))

(setq rowOffset (+ rowOffset rowOffsetOrig)))))

(return-from get-number-in-a-row total))

(defun get-playable-spaces (column row columnOffsetOrig rowOffsetOrig)

(setq continue 1)

(setq total 0)

(setq columnOffset columnOffsetOrig)

(setq rowOffset rowOffsetOrig)

(loop while (= continue 1) do

(setq thisPiece (get-piece (+ column columnOffset) (+ row rowOffset)))

(setq belowPiece (get-piece (+ column columnOffset) (+ row rowOffset -1)))

(if (or (string/= thisPiece "\_") (string= belowPiece "\_"))

(setq continue 0)

(progn

(setq total (+ total 1))

(setq columnOffset (+ columnOffset columnOffsetOrig))

(setq rowOffset (+ rowOffset rowOffsetOrig)))))

(return-from get-playable-spaces total))

(defun get-score (total rightSpaces leftSpaces player)

(if (> total 3)

(if (string= player "O")

(return-from get-score 50000)(return-from get-score 10000)))

(if (and (= total 3) (> rightSpaces 0)(> leftSpaces 0))

(if (string= player "O")

(return-from get-score 5000)(return-from get-score 1000)))

(if (= total 3)

(return-from get-score 100))

(if (and (= total 2) (> rightSpaces 0)(> leftSpaces 0) (or (> rightSpaces 1)(> leftSpaces 1)))

(return-from get-score 10))

(if (= total 2)

(return-from get-score 1)))

(defun check-in-a-row (column row player)

(setq score 0)

"check horizontal"

(setq rightTotal (get-number-in-a-row column row 1 0 player))

(setq rightSpaces (get-number-in-a-row (+ column rightTotal) row 1 0 "\_"))

(setq rightPlayableSpaces (get-playable-spaces (+ column rightTotal) row 1 0))

(setq leftTotal (get-number-in-a-row column row -1 0 player))

(setq leftSpaces (get-number-in-a-row (- column leftTotal) row -1 0 "\_"))

(setq leftPlayableSpaces (get-playable-spaces (- column leftTotal) row -1 0))

(if (and (> (+ rightTotal leftTotal) 0)(> (+ rightTotal leftTotal rightSpaces leftSpaces) 2))

(setq score (+ score (get-score (+ rightTotal leftTotal 1) rightPlayableSpaces leftPlayableSpaces player))))

"check vertical"

(setq total (get-number-in-a-row column row 0 -1 player))

(setq spaces (- 7 (nth column columnPieces)))

(if (and (> total 0)(> (+ total spaces) 2))

(setq score (+ score (get-score (+ total 1) 0 0 player))))

"check down diaganol"

(setq rightTotal (get-number-in-a-row column row 1 -1 player))

(setq rightSpaces (get-number-in-a-row (+ column rightTotal) (- row rightTotal) 1 -1 "\_"))

(setq rightPlayableSpaces (get-playable-spaces (+ column rightTotal) (- row rightTotal) 1 -1))

(setq leftTotal (get-number-in-a-row column row -1 1 player))

(setq leftSpaces (get-number-in-a-row (- column leftTotal) (+ row leftTotal) -1 1 "\_"))

(setq leftPlayableSpaces (get-playable-spaces (- column leftTotal) (+ row leftTotal) -1 -1))

(if (and (> (+ rightTotal leftTotal) 0)(> (+ rightTotal leftTotal rightSpaces leftSpaces) 2))

(setq score (+ score (get-score (+ rightTotal leftTotal 1) rightPlayableSpaces leftPlayableSpaces player))))

"check up diaganol"

(setq rightTotal (get-number-in-a-row column row 1 1 player))

(setq rightSpaces (get-number-in-a-row (+ column rightTotal) (+ row rightTotal) 1 1 "\_"))

(setq rightPlayableSpaces (get-playable-spaces (+ column rightTotal) (+ row rightTotal) 1 1))

(setq leftTotal (get-number-in-a-row column row -1 -1 player))

(setq leftSpaces (get-number-in-a-row (- column rightTotal) (- row rightTotal) -1 -1 "\_"))

(setq leftPlayableSpaces (get-playable-spaces (- column leftTotal) (- row rightTotal) -1 -1))

(if (and (> (+ rightTotal leftTotal) 0)(> (+ rightTotal leftTotal rightSpaces leftSpaces) 2))

(setq score (+ score (get-score (+ rightTotal leftTotal 1) rightPlayableSpaces leftPlayableSpaces player))))

(return-from check-in-a-row score))

(defun check-board-full ()

(do ((i 0 (+ i 1)))((= i 8) 'done)

(if (< (nth i columnPieces) 8)

(return-from check-board-full 0)))

(return-from check-board-full 1))

(defun player-move()

(setq column (get-choice))

(if (= column -1)

(return-from player-move 1))

(setq row (nth column columnPieces))

(setf (nth column (nth row gameBoard)) "X")

(setf (nth column columnPieces) (+ row 1))

(if (> (check-in-a-row column row "X") 9999)

(progn

(show)

(format t "~%You win!!")

(return-from player-move 1)))

(return-from player-move 0))

(defun computer-move()

(setq column (get-computer-move))

(if (= column -1)

(return-from computer-move 1))

(setq row (nth column columnPieces))

(setf (nth column (nth row gameBoard)) "O")

(setf (nth column columnPieces) (+ row 1))

(show)

(format t "~%Computer plays column ~A" (+ 1 column))

(if (> (check-in-a-row column row "O") 9999)

(progn

(format t "~%Computer wins :-(")

(return-from computer-move 1)))

(if (= 1 (check-board-full))

(progn

(format t "~%Board is full, game ends in a tie")

(return-from computer-move 1)))

(return-from computer-move 0))

(defun play()

(setq gameOver 0)

(show)

(loop while (= gameOver 0) do

(setq gameOver (player-move))

(if (= gameOver 0)

(setq gameOver (computer-move)))))