Task 1

Single Reflex agent

```
import random


# initialize the tic-tac-toe board
board = [' ' for x in range(9)]


# function to print the board
def print_board():
    print('   |   |')
    print(' ' + board[0] + ' | ' + board[1] + ' | ' + board[2])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[3] + ' | ' + board[4] + ' | ' + board[5])
    print('   |   |')
    print('-----------')
    print('   |   |')
    print(' ' + board[6] + ' | ' + board[7] + ' | ' + board[8])
    print('   |   |')


# function to check if the board is full
def is_board_full(board):
    if board.count(' ') > 1:
        return False
    else:
        return True


# function to check for a win
```

```
def check_win(board, player):
    if (board[0] == player and board[1] == player and board[2] == player) or \
       (board[3] == player and board[4] == player and board[5] == player) or \
       (board[6] == player and board[7] == player and board[8] == player) or \
       (board[0] == player and board[3] == player and board[6] == player) or \
       (board[1] == player and board[4] == player and board[7] == player) or \
       (board[2] == player and board[5] == player and board[8] == player) or \
       (board[0] == player and board[4] == player and board[8] == player) or \
       (board[2] == player and board[4] == player and board[6] == player):
        return True
    else:
        return False


# function for a simple reflex agent's move
def simple_reflex_agent(board, player):
    # check if the agent can win
    for i in range(9):
        if board[i] == ' ':
            board[i] = player
            if check_win(board, player):
                return i
            else:
                board[i] = ' '

    # check if the opponent can win
    opponent = 'O' if player == 'X' else 'X'
    for i in range(9):
        if board[i] == ' ':
            board[i] = opponent
            if check_win(board, opponent):
```

```
            board[i] = player
            return i
        else:
            board[i] = ' '

    # make a random move
    while True:
        move = random.randint(0,8)
        if board[move] == ' ':
            return move

# function for an Agent vs Agent match
def agent_vs_agent():
    # initialize the players
    player1 = 'X'
    player2 = 'O'

    # randomly decide which player goes first
    current_player = random.choice([player1, player2])

    # play the game
    while not is_board_full(board):
        if current_player == player1:
            # player 1's turn
            print("Player 1 (", player1, ") makes a move.")
            move = simple_reflex_agent(board, current_player)
            board[move] = current_player
        else:
        # player 2's turn
            print("Player 2 (", player2, ") makes a move.")
```

```
            move = simple_reflex_agent(board, current_player)
            board[move] = current_player

        # print the current board
        print_board()

        # check for a win
        if check_win(board, current_player):
            print("Player", current_player, "wins!")
            return

        # switch to the other player
        current_player = player2 if current_player == player1 else player1

# if the board is full and no one has won, it's a tie
print("It's a tie!")
```

Lookup Table:

```
lookup_table = {'0': '-', '1': 'X', '2': 'O'}
# Define the Tic Tac Toe board as a list of lists
board = [['0', '0', '0'], ['0', '0', '0'], ['0', '0', '0']]

# Define the function to print the board
def print_board():
    for row in board:
        print('|'.join([lookup_table[col] for col in row]))

# Define the function to check if a player has won
def check_win(player):
    for i in range(3):
        # Check horizontal lines
        if board[i][0] == board[i][1] == board[i][2] == player:
            return True
        # Check vertical lines
        if board[0][i] == board[1][i] == board[2][i] == player:
            return True
    # Check diagonal lines
    if board[0][0] == board[1][1] == board[2][2] == player:
        return True
    if board[0][2] == board[1][1] == board[2][0] == player:
        return True
    return False
```

```
# Define the main game loop
current_player = '1'
while True:
    # Print the board
    print_board()

    # Get the user input for the next move
    row = int(input("Enter row number (1-3): ")) - 1
    col = int(input("Enter column number (1-3): ")) - 1

    # Check if the chosen cell is empty
    if board[row][col] != '0':
        print("This cell is already occupied. Try again.")
        continue

    # Update the board with the current player's move
    board[row][col] = current_player

    # Check if the current player has won
    if check_win(current_player):
        print_board()
        print("Player " + current_player + " wins!")
        break

    # Check if the board is full
    if all([all(row) for row in board]):
        print_board()
        print("The game is a tie!")
```

```
        break

    # Switch to the other player
    current_player = '2' if current_player == '1' else '1'
```

Task 2

```
 import random

# Define the lookup table for Tic Tac Toe game
lookup_table = {'0': '-', '1': 'X', '2': 'O'}

# Define the Tic Tac Toe board as a list of lists
board = [['0', '0', '0'], ['0', '0', '0'], ['0', '0', '0']]

# Define the function to print the board
def print_board():
    for row in board:
        print('|'.join([lookup_table[col] for col in row]))

# Define the function to check if a player has won
def check_win(player):
    for i in range(3):
        # Check horizontal lines
        if board[i][0] == board[i][1] == board[i][2] == player:
            return True
        # Check vertical lines
        if board[0][i] == board[1][i] == board[2][i] == player:
            return True
    # Check diagonal lines
```

```
    if board[0][0] == board[1][1] == board[2][2] == player:
        return True
    if board[0][2] == board[1][1] == board[2][0] == player:
        return True
    return False

# Define the function for the computer's move
def computer_move():
    # Look for a winning move
    for i in range(3):
        for j in range(3):
            if board[i][j] == '0':
                board[i][j] = '2'
                if check_win('2'):
                    return
                board[i][j] = '0'

    # Look for a blocking move
    for i in range(3):
        for j in range(3):
            if board[i][j] == '0':
                board[i][j] = '1'
                if check_win('1'):
                    board[i][j] = '2'
                    return
                board[i][j] = '0'

    # Choose a random move
    while True:
```

```
        i, j = random.randint(0, 2), random.randint(0, 2)
        if board[i][j] == '0':
            board[i][j] = '2'
            return

# Define the main game loop
current_player = random.choice(['1', '2'])
print("The game has started. The current player is " + lookup_table[current_player])

while True:
    if current_player == '1':
        # Player's move
        print_board()
        row = int(input("Enter row number (1-3): ")) - 1
        col = int(input("Enter column number (1-3): ")) - 1
        if board[row][col] != '0':
            print("This cell is already occupied. Try again.")
            continue
        board[row][col] = '1'
        if check_win('1'):
            print_board()
            print("You win!")
            break
    else:
        # Computer's move
        print("Computer's turn:")
        computer_move()
        if check_win('2'):
            print_board()
```

```
        print("The computer wins!")
        break


    if all([all(row) for row in board]):
        print_board()
        print("The game is a tie!")
        break


    # Switch to the other player
    current_player = '2' if current_player == '1' else '1'
```

```
The game has started. The current player is X
-|-|-
-|-|-
-|-|-
Enter row number (1-3): 2
Enter column number (1-3): 1
-|-|-
X|-|-
-|-|-
The game is a tie!
```

Task 3

```
# Tic Tac Toe single reflex agent with 4-cell combinations


table = {(0, 1, 2): 1, (3, 4, 5): 1, (6, 7, 8): 1,  # horizontal
        (0, 3, 6): 1, (1, 4, 7): 1, (2, 5, 8): 1,  # vertical
        (0, 4, 8): 1, (2, 4, 6): 1}             # diagonal


def check_winner(board):
    for indices, _ in table.items():
        if board[indices[0]] == board[indices[1]] == board[indices[2]] != 0:
            return board[indices[0]]
    if 0 not in board:
        return 0
    return None


def get_score(board, player, cell_indices):
    score = 0
    for indices, p in table.items():
        if all(board[cell_indices[i]] == player for i in range(len(cell_indices))) and \
           all(board[indices[i]] == player for i in range(len(indices)) if i not in cell_indices):
            score += p
```

```
        elif any(board[cell_indices[i]] == -player for i in range(len(cell_indices))) and \
             any(board[indices[i]] == -player for i in range(len(indices)) if i not in cell_indices):
            score -= p
    return score


def get_best_move(board, player, cell_indices):
    best_score = -float("inf")
    best_move = None
    for i in range(len(board)):
        if board[i] == 0:
            board[i] = player
            score = get_score(board, player, cell_indices)
            if score > best_score:
                best_score = score
                best_move = i
            board[i] = 0
    return best_move


board = [0] * 9
player = 1
cell_indices = [0, 1, 3, 4]  # change this to the indices of the 4 cells you have selected
while True:
    print("".join(["X" if i == 1 else "O" if i == -1 else "-" for i in board]))
    if player == 1:
        move = int(input("Enter move: "))
    else:
        move = get_best_move(board, player, cell_indices)
    if board[move] != 0:
        print("Invalid move")
```

```
    continue
board[move] = player
winner = check_winner(board)
if winner != None:
    print("".join(["X" if i == 1 else "O" if i == -1 else "-" for i in board]))
    if winner == 0:
        print("Tie!")
    else:
        print("Player", "X" if winner == 1 else "O", "wins!")
    break
player *= -1
```

```
---------
Enter move: 1
-X-------
OX-------
Enter move: 2
OXX------
OXXO-----
Enter move: 3
Invalid move
OXXO-----
Enter move: 4
OXXOX----
OXXOXO---
Enter move: 5
Invalid move
OXXOXO---
Enter move: 6
OXXOXOX--
Player X wins!
```