# Missouri State University
## Department of Computer Science

Assignment 1: Evolutionary Computing
Summer 2025

## Submission Instructions

Students must follow these submission guidelines to receive full credit:

1. **Submission:**
   - Submit a single compressed '.zip' file containing all source files (Python scripts) and PDF report for conceptual questions.
2. **File Naming Convention:** Name your '.zip' file as: `FirstName_LastName_EC_AS1.zip`
3. **Deadline:** Submit your assignment by June 16$^{th}$ 2025.
4. **Academic Integrity:**
   - Your submission must be your own work.
   - Plagiarism will result in a zero grade and possible disciplinary action.

## Objective

This assignment aims to develop your understanding of local search algorithms by implementing the **Hill Climbing** technique to solve the classic **8-Puzzle Problem**. You will write code to generate puzzle instances, compute heuristic costs, generate neighbor states, and implement the Hill Climbing algorithm using a given visualization function.

## 1. Problem Statement

The 8-puzzle consists of a 3×3 board with tiles numbered from 1 to 8 and one blank tile. The objective is to reach the goal state:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & \end{bmatrix}$$

You are provided with a function to visualize a puzzle board and a function to generate a random 8-puzzle instance from the goal state by applying a series of valid random moves. You are required to implement all other necessary components to solve the puzzle using Hill Climbing.

## Tasks

### Q1. Implementation Tasks

Implement the following functions in Python:

1. A heuristic function that calculates the number of misplaced tiles.
2. A function to generate all valid neighbor states from the current configuration.
3. The Hill Climbing algorithm using the above components to solve the puzzle.
4. Use the provided `visualize_8_puzzle()` function to display the puzzle state at each step.

### Q2. Evaluation and Observation

1. Run your code with at least 3 different initial states and record the output.
2. Capture and include screenshots or plots for:
   - One run that reaches the goal state.
   - One run that gets stuck in a local minimum.
3. Briefly explain the behavior of your algorithm in both cases.

### Q3. Conceptual Questions

1. What are the main limitations of the Hill Climbing approach in this problem?

## 2. Problem Description

The 8-Queens problem requires placing 8 queens on an 8×8 chessboard so that no two queens threaten each other. This means no two queens may share the same row, column, or diagonal. You are provided with a Python function that visualizes the current board state and its associated cost (number of attacking queen pairs). You must implement the remaining components of the algorithm.

## Tasks

### Q1. Implementation Tasks

Implement the following components in Python:

1. A function to generate a random 8-Queens state represented by a list of 8 integers, where the $i$-th value indicates the row of the queen in column $i$.
2. A heuristic function that computes the number of pairs of queens that are attacking each other.
3. A function to generate possible neighbors of the current state by changing the position of one queen at a time.
4. The Hill Climbing algorithm to search for a non-attacking configuration.
5. Use the provided `visualize_queens()` function to visualize the board at each step.

### Q2. Evaluation

1. Run your code with at least 5 different random initial states. For each run, report:
   - The initial state and its heuristic value.
   - The final state and its heuristic value.
2. Include at least one case where the algorithm gets stuck in a local minimum and does not find a complete solution.

### Q3. Conceptual Questions

1. Why does Hill Climbing sometimes get stuck in local minima in the 8-Queens problem?
2. Explain how random restarts help mitigate this issue.