Course: **Artificial Intelligence**	Assignment 02	**DUE DATE:** 10/04/2025
Course Code: **AI2001**		Total Questions: 6

Instructions:

1. Attempt all questions.
2. Submit a pdf for the **hand-written** assignments and python (or any language of your own choice) code file for programming questions.

Question 1:

You're designing an autonomous drone for a hiking mission. This drone is equipped with a sensor that can measure the elevation (height) at its current position, but it cannot see the entire terrain at once. You aim to program the drone to find the highest peak in a given one-dimensional mountain range. The elevation increases up to a peak and then decreases; this forms an unimodal pattern. To simulate this task, you're given access to a black-box function query(x) that returns the elevation at position x, where x is an integer between 0 and N. You are given:

A function

```
def query(x: int) -> int
```

returns the elevation at index x.

An integer N represents the maximum value of x. You need to implement the function

```
def find_peak(N: int) -> int
```

that uses a Hill Climbing algorithm to find the index p ($0 \le p \le N$) at which the elevation is maximum.

The function query(x) follows the unimodal rule:

There exists a peak index p such that:

```
query(x) < query(x + 1) for all x < p
query(x) > query(x + 1) for all x ≥ p
```

SUPPOSE a query function is this can be replaceable with any function.

```
def query(x):
    return -1 * (x - 7)**2 + 49
```


Question 2: Optimizing a Multi-stage Manufacturing Process Using Genetic Algorithms

You are part of a team at a large manufacturing company producing electronic devices. The company operates 4 production stages (e.g., assembly, testing, packaging, and shipping) at 3 production facilities located in different regions. The company needs to optimize its production process by efficiently allocating tasks at each stage to each facility.

The company wants to minimize the total production time and costs while satisfying the following constraints:

- Production Time Constraint: Each facility can only process a certain number of tasks in a given day due to its available working hours.
- Task Constraints: Each production task requires a different amount of time, and each facility has different costs associated with performing the tasks.

The goal is to allocate tasks to facilities in such a way that the total production time and cost are minimized while ensuring that no facility exceeds its capacity.

Parameters:

- **Production Tasks and Times (in hours):**

Task	Time Required (hrs)
Task 1	5
Task 2	8
Task 3	4
Task 4	7
Task 5	6
Task 6	3
Task 7	9

- **Production Facilities and Their Capacities (in hours per day):**

Facility	Capacity (hrs/day)
Facility 1	24
Facility 2	30

Facility 3	28

- **Cost Matrix (cost per hour for each task at each facility):**

Task	Facility 1	Facility 2	Facility 3
Task 1	10	12	9
Task 2	15	14	16
Task 3	8	9	7
Task 4	12	10	13
Task 5	14	13	12
Task 6	9	8	10
Task 7	11	12	13

Optimization Goal:

Minimize the total production time and costs while ensuring:

- Each task is assigned to one and only one facility.
- No facility exceeds its capacity of hours per day.

Constraints:

1. Production time constraint: The total time allocated to a facility's tasks should not exceed its daily capacity.
2. Cost constraint: Production costs should be minimized by efficiently assigning tasks to the facilities with the lowest cost for each task.

Genetic Algorithm Setup:

- Population size: 6 chromosomes
- Crossover rate: 80%
- Mutation rate: 20%
- Selection method: Roulette Wheel Selection
- Crossover method: One-point crossover
- Mutation method: Swap mutation (swap allocations between warehouses and stores) ● Fitness function: The fitness will be based on the total transportation cost. If the capacity constraints are violated, the fitness value will be penalized.

Note: Solve the question in a Dry-Run way that includes all needed steps and a coded solution in any language you choose.

Question 3

A Sudoku puzzle is a 9x9 grid (81 variables) where each cell in the grid can take on the integer values 1-9 (the domain of each variable). A solution to a Sudoku puzzle is an assignment of values for each cell in the grid such that no two cells in the same row, column, or 3x3 square have the same value.

For example, given an initial configuration of a Sudoku puzzle, the corresponding solution could be:

```
483|921|657          ..3|.2.|6..
967|345|821          9..|3.5|..1
251|876|493          ..1|8.6|4..
---------- ----------
548|132|976          ..8|1.2|9..
729|564|138          7..|...|..8
136|798|245          ..6|7.8|2..
---------- ----------
372|689|514          ..2|6.9|5..
814|253|769          8..|2.3|..9
695|417|382          ..5|.1.|3..
```

Initial Configuration Solution

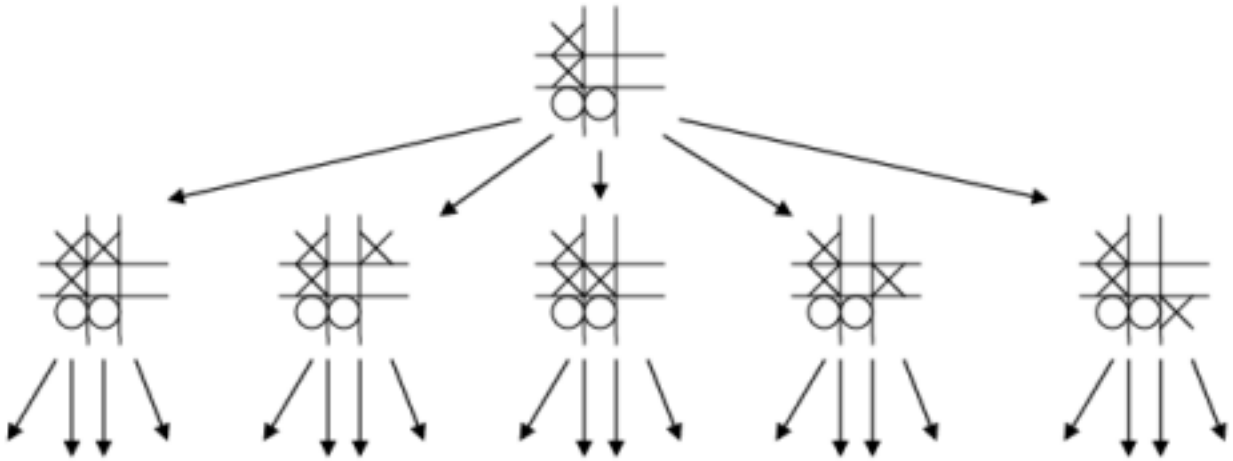
- Your task is to write a program that can take a set of Sudoku puzzles as input from a file, model each puzzle as a CSP, and show output of the solution to each puzzle. You should make use of both Constraint Propagation (i.e., arc consistency AC3) and Backtracking Search algorithms as part of your solution.
- Your program should be able to read in these puzzles, solve them, then output the solutions in the same format (a string of 81 digits, followed by a newline character) in the same order they were read in from the file.
- You should not import any modules or libraries not already built into your language of choice, nor should you use any code from online sources.
- Now implement the Sudoku Puzzle in Google OR tools you have studied in the lab. ● Get an implementation of Sudoku Puzzle online from GITHUB or ask ChatGPT to build one for you. ● How is your version different from the online (or Chat GPT one.)
- Now time all the three versions (1. Yours, 2. Google OR Tools, 3. Github(or ChatGPT). Show time comparisons.
- Can you improve time on your version?

Question 4

You are the X player, looking at the board shown below, with five possible moves. You want to look ahead to find your best move and decide to use the following evaluation function for rating board configurations:

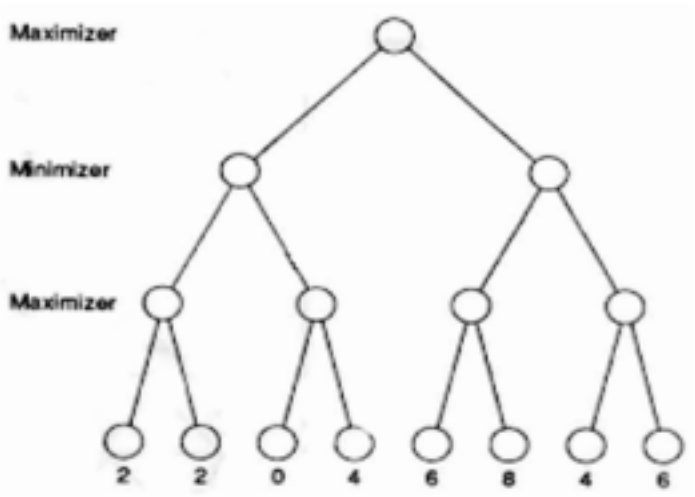
```
value V = 0
 do over all rows, columns, diagonals R:
     if R contains three Xs, V = 1000
     else if R contains three Os, V = -1000
            else when R contains only two Xs, V = V + 100
            else when R contains only one X, V = V + 10
            else when R contains only two Os, V = V – 100
            else when R contains only one O, V = V – 10
  end do
return V
```

Draw the four configurations possible from the leftmost and rightmost board configurations below. Use the above static evaluation function to rate the 8 board configurations and choose X's best move. (A reminder: The board configurations that you draw will show possibilities for O's next move.)

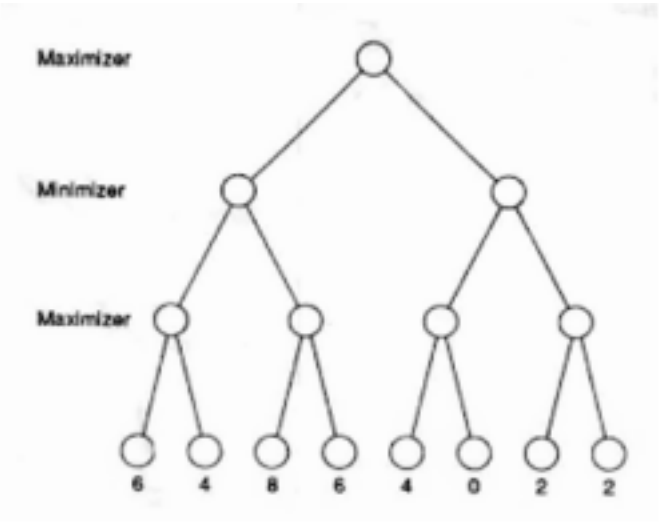


Question 5:

A. Consider the game tree shown below. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off, and indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



B. Now consider the tree shown below. which is a mirror image of the tree shown above. Explore the tree using the alpha-beta procedure. Indicate all parts of the tree that are cut off. Indicate the winning path or paths. Strike out all static evaluation values that do not need to be computed.



Question 6:

A company uses an AI-powered Intrusion Detecon System (IDS) to defend its network against cyberaacks. The system (Max) and the aacker (Min) engage in a turn-based adversarial game, with the aacker trying to breach security and the AI trying to defend. Some aacks have probabilisc outcomes due to unknown vulnerabilies.

Part (a): Game Model

Describe the following:

1. Players: Who are the players in this game, and what are their objecves?

2. Decision-Making: How do each of the players make their decisions in this adversarial setup? 3. Stochasc Elements: How do probabilisc aacks (e.g., zero-day exploits) impact the defender's strategy?

Part (b): Construcng the Game Tree

Consider the following acons:

- Max (Defender):
    - Deploy Firewall: Strong security, high cost.
    - Patch System: Fix vulnerabilies; zero-day exploits are sll possible.
    - Ignore Alerts: Risky, allows potenal real aacks to succeed.
- Min (Aacker):
    - Brute Force Aack: Easily detected, predictable outcome.
    - Phishing Aack: Tricky, moderate risk.
    - Zero-Day Exploit: 50% chance of success.
    - Fake Aack: Misleading alert, wastes resources.
    - Real Aack: If ignored, it succeeds completely.

Task:

- Draw the game tree with these acons, showing stochasc nodes where applicable (e.g., Zero-Day Exploit).

Part (c): Minimax and Alpha-Beta Pruning

1. Using Minimax, determine the best move for the Defender (Max), assuming the Aacker (Min) picks the move that causes the most damage to the Defender.
2. Apply Alpha-Beta Pruning to your game tree and indicate which branches can be pruned.

Part (d): Expecmax and Probabilisc Aacks

1. Calculate the expected value of the zero-day exploit with a 50% success rate. 2. How should the Defender (Max) adjust its strategy if it switches to using Expecmax instead of Minimax?