

Programming Challenges:

1 - Challenge: Binary Code Decoding

Challenge Objective:

In this challenge, you need to design a JavaScript program that can decode a binary string (composed of 0s and 1s) back into its original text. To do this, you must convert the binary codes into their corresponding characters or words based on the provided table, where each letter has a specific binary code assigned.

Key Notes:

- ✓ **Prefix-Free Encoding:** No code is a prefix of another, ensuring unambiguous decoding.
- ✓ **Shorter Codes for Common Letters:** Frequent letters in English, such as E, T, A, O, I, N, have shorter binary codes.

A	1101
B	10110
C	0001
D	0110
E	111
F	00101
G	01001
H	00001
I	1010
J	010000
K	10001
L	0111
M	11000
N	1001
O	0011
P	11101
Q	11001
R	10000
S	01101
T	0101
U	10111
V	000001
W	10101
X	001001
Y	0100001
Z	11100
SPACE	00000

- ✓ This table ensures no prefix conflicts, making the decoding process error-free.



Your program must correctly decode the following binary sequence:


```
10101 111 0111 0001 0011 11000 111 0101 0011 0101 00001 111 0001
0011 11000 11101 111 0101 1010 0101 1010 0011 1001 0011 1001 111
1010 0110 111 1101 0011 1001 111 1010 0110 111 1101 0011 1001
10101 0011 10000 0111 0110 0101 10111 10000 10001 111 0100001
```

Participant Responsibilities:

- Implement the binary decoding algorithm using JavaScript
- Accurately convert binary codes to corresponding characters using the provided table
- Preserve spaces between words for better clarity and readability
- Ensure full accuracy in converting codes to correct letters and words

Your JavaScript program should return:

- Welcome to the competition, One Idea, One World, Turkey.

 **Note:** This challenge will test your JavaScript skills, enhance your understanding of binary decoding, and ultimately reward you with a hidden message once you successfully decode it!



2 - Challenge: Intelligent Path Finder in a City Network

Challenge Objective:

Your task is to design an intelligent system that can find the shortest path between two points in a city map. To implement the map, you may use map-reading APIs such as Google Maps or similar services. The system should allow the user to select two points on the map, compute and display possible routes with specified travel times.

Additionally, the system should be designed to eliminate longer routes and, in the end, suggest the optimal path based on travel time.

Requirements:

- ✓ **Implementation using HTML, CSS, and JavaScript** to create an interactive user interface.
- ✓ **Use of Dijkstra's algorithm or A*** to find the shortest path between two points.
- ✓ **Use of map-reading APIs** to display the map and routes (such as Google Maps or similar).
- ✓ **Design a visual interface** that allows users to view routes and travel times.
- ✓ **Eliminate longer routes** and display the optimal path with the shortest travel time.

Participant Responsibilities:

- **Develop a web application:**
 - Use HTML, CSS, and JavaScript to build an interactive user interface.
 - Optimize user interactions for selecting points and displaying the optimal path.
- **Implement pathfinding algorithms:**
 - Use Dijkstra's algorithm or A* to find the shortest path.
 - Update the path calculations in real-time.
- **Use map-reading APIs:**
 - Use Google Maps API or similar services to display the map and calculate the route.
 - Show the possible routes with specified travel times to the user.
- **Eliminate longer routes:**
 - Identify and eliminate longer routes, presenting only the optimal path with shorter travel time.

 **Note:** This challenge will enhance your JavaScript skills and improve your understanding of pathfinding algorithms and urban navigation systems!



3 - Challenge: Simulating Machinery Tracking System in a Factory with C++

Challenge Objective:

Design a C++ program capable of simulating a machinery tracking system for a factory. This program should be able to store and report machinery statuses, maintenance logs, and operational times.

Challenge Description: In this challenge, you need to design a simple system for tracking machinery in a factory, including the following features:

- 1. Registering Machines:** Each machine should be registered with attributes like ID, name, status (active or under repair), and operational times.
- 2. Updating Machine Status:** The system should allow the operator to manually update the status of each machine (e.g., changing from "active" to "under repair").
- 3. Logging Repairs:** Whenever a machine goes under repair, the start and end times of the repair should be logged.
- 4. Generating Reports:** The system should generate reports showing the current status of each machine, operational hours, and the number of repairs.

Requirements:


- 1. ✓ Register Machines:** You should define a class for machines that includes attributes like ID, name, status, and operational times.
- 2. ✓ Updating Status:** The system should allow changing the machine status from "active" to "under repair" and vice versa.
- 3. ✓ Generating Reports:** The program should display the current status of each machine, along with their operational hours.

1. Input:

- Machine ID
- Machine Name
- Status (Active or Under Repair)
- Repair times

2. Output:

- Machine status report
- Operational hours and repair logs

 **Note:** This challenge will help you strengthen your C++ programming skills while designing a real-world management system.



4 - Challenge: Smart Home Energy Management System

Challenge Objective:

Design a system that monitors and optimizes energy consumption in a smart home. This system should be able to turn devices on or off based on time of day, electricity usage, and consumption limits, while also displaying real-time energy consumption.

Requirements:

- ✓ Implement using HTML, CSS, and JavaScript to create an interactive UI.
- ✓ Display a list of household devices (e.g., lights, refrigerator, air conditioner, TV, etc.).
- ✓ Manage energy consumption and suggest turning off high-energy devices during peak hours.
- ✓ Show real-time energy usage and a graph of daily consumption.
- ✓ Enable an energy-saving mode that automatically reduces power consumption.

Expected Input & Output:

Input:

- A list of household electrical devices and their power consumption rates.
- Different times of the day and electricity tariff categories (peak hours vs. off-peak hours).
- A maximum acceptable energy threshold to prevent power outages.

Output:

- Real-time power usage and a graph displaying daily energy consumption.
- Suggestions to turn off unnecessary devices during peak hours.
- Energy-saving mode, which automatically shuts down selected devices to reduce consumption.

Participants' Responsibilities:

◆ Develop a web application:


- Create an interactive UI that displays devices and their energy usage.
- Allow users to toggle devices on/off with a single click.

◆ Implement an energy management algorithm:

- Identify high-consumption devices and prioritize them for shutdown.
- Suggest energy-saving strategies without disrupting home functionality.

◆ Display output data effectively:

- Show a graph of energy consumption throughout the day.
- Display warnings such as "Excessive Consumption!" when approaching the limit.

 **Note:** This challenge will test your skills in resource management, UI/UX design, and JavaScript-based energy optimization algorithms!



Artificial Intelligence Challenges:

1 - Text Sentiment Analysis System

Challenge Objective:

In this challenge, you need to design an artificial intelligence model that can analyze and detect the sentiment of input texts. The system should be able to classify sentences into three categories: positive, negative, and neutral, and it should also display the accuracy of its analysis. This challenge will focus on sentiment analysis in written text using Natural Language Processing (NLP) techniques and machine learning.

Requirements:

- ✓ Implementation in Python using TensorFlow or PyTorch
- ✓ Use of Natural Language Processing (NLP) models

Expected Output:

Input: A sentence provided by the user as input.

Output: The model should classify the sentiment of the sentence into one of the categories: positive, negative, or neutral, and display the probability percentage for each (e.g., 70% positive, 20% negative, 10% neutral).

Participant Tasks:

- **Design the Sentiment Analysis Model:**


- Use Python with TensorFlow or PyTorch to implement the model.
- The model must be capable of classifying sentiments into three **categories: positive, negative, and neutral.**

- **Train the Model with Text Data:**

- Use publicly available text data to train the model so that it can identify sentiments in different sentences.
- The model should continuously learn and improve its accuracy after each prediction.

- **Display Sentiment Analysis Results:**

- After each prediction, the model should display its accuracy.
- The output should include the sentiment classification along with a confidence level percentage.

 **Note:** This challenge will help you improve your skills in Natural Language Processing (NLP) and machine learning, as well as your ability to evaluate model accuracy.



2 - Disease Detection via Voice

Challenge Objective:

In this challenge, you need to design an artificial intelligence model that can analyze a user's voice to detect symptoms of respiratory diseases, such as COVID-19. The system should process audio samples and display the likelihood of the user having the disease based on the analysis of acoustic features. This task involves voice processing and applying machine learning techniques for health detection..

Requirements:

- ✓ Training the model with available public audio datasets for disease detection
- ✓ Use of Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) for audio feature analysis
- ✓ The system should provide a probability score indicating the likelihood of disease presence based on voice analysis


Expected Output:

Input: An audio file provided by the user (e.g., a voice recording).

Output: The system should display the percentage probability of the user having respiratory diseases like COVID-19 based on the audio sample. The output should also include the key audio features (such as pitch, tone, etc.) and the significance of each feature in the final decision-making process.

Participant Tasks:

- **Design the Disease Detection Model:**
 - Implement the model using Python and librosa for sound processing.
 - The model should analyze voice recordings to detect respiratory disease symptoms.
- **Train the Model with Audio Data:**
 - Use publicly available audio datasets to train the model so that it can detect the presence of disease from voice samples.
 - The model should continuously improve its prediction accuracy by learning from the data.
- **Extract and Analyze Audio Features:**
 - The model should extract acoustic features from the input audio file, such as pitch, rhythm, and frequency patterns, and use them to make predictions.

 **Note:** This challenge helps develop your skills in audio analysis, machine learning, and healthcare AI, focusing on real-world applications for detecting diseases through voice recognition.



3 - Challenge: AI Chatbot Simulating Atatürk's Personality

Challenge Objective:

Design an AI-powered chatbot that emulates the personality, speech style, and ideologies of Mustafa Kemal Atatürk. The chatbot should provide historically accurate responses as if Atatürk himself were answering questions. The goal is to bring historical figures to life using AI-driven natural language processing (NLP) and an interactive user interface.

Challenge Objective:

In this challenge, you need to develop a chatbot that can:

1. **Understand and process user questions** about history, politics, leadership, and Atatürk's ideologies.
2. **Respond in Atatürk's voice and personality**, using formal yet inspiring language.
3. **Have a beautiful and intuitive user interface** to make conversations seamless and engaging.

To achieve this, participants will need to:

- ✓ Train a Natural Language Processing (NLP) model based on historical speeches, books, and writings by and about Atatürk.
- ✓ Design a web-based or desktop UI that allows users to chat with the bot interactively.
- ✓ Implement response filtering to ensure the chatbot remains historically accurate and appropriate.
- ✓ Optionally, add speech synthesis (TTS) to make Atatürk's voice part of the experience.

Key Features & Functionalities:

Historical Accuracy: The chatbot must provide factual, insightful, and ideologically consistent responses.

Conversational Memory: Maintain context in multi-turn conversations.

Interactive UI: Build an elegant, responsive web or mobile chat interface.

Voice Output (Optional): Generate responses using speech synthesis to make it more immersive.

Input & Output Examples:

🗨️ **User:** What is your vision for Turkey's future?

🤖 **Chatbot (Atatürk):** "My vision for Turkey is a modern, secular, and progressive nation where education, science, and democracy thrive. Every citizen must contribute to its advancement!"

📢 **Note:** This challenge will test your AI, NLP, UI/UX, and software development skills while bringing history to life through modern technology!



4 - Challenge: Intelligent Hunter with Reinforcement Learning

Challenge Objective:

In this challenge, you need to design an AI agent (Hunter) that uses Reinforcement Learning (RL) to improve its hunting skills over time. The agent must navigate through a simulated environment to catch a prey, while the prey attempts to escape. Initially, the hunter has no prior knowledge, but it learns optimal strategies using Stable Baselines3 and reinforcement learning algorithms. The prey follows a basic escape strategy, but as the hunter improves, the challenge becomes harder.

Challenge Objective:

- ✓ Implement a hunter agent using Stable Baselines3 and reinforcement learning techniques.
- ✓ Train the agent to maximize its hunting success rate.
- ✓ Evaluate the agent's performance based on its ability to catch the prey efficiently.

Environment Setup & Rules:

- Simulated Environment: A 2D grid-based world (e.g., GridWorld or OpenAI Gym environment).
- Movement:
 - Both hunter and prey can move in four directions (up, down, left, right).
 - The prey moves randomly at first but may develop evasive patterns over **time**.
- Hunter's Learning Process:
 - The hunter starts with no knowledge and learns from trial and error.
 - It must optimize its movements to capture the prey faster.
- Rewards & Penalties:
 - The hunter receives a positive reward when it catches the prey.
 - The hunter gets a negative reward (penalty) if the prey escapes.
 - Moving towards the prey grants small rewards to encourage learning.

Expected Output:

● **Input:** A simulated environment where the hunter and prey move dynamically.

● **Output:** The hunter agent progressively learns and improves its strategy for capturing the prey.

📊 **Performance Metrics:** The system should display success rates (e.g., percentage of successful hunts vs. failed attempts).



Why This Challenge?

🎯 **Develop hands-on experience** with Reinforcement Learning and Stable Baselines3.

🎯 **Learn key RL concepts** such as **Policy Optimization**, Reward Engineering, and Exploration vs. Exploitation.

🎯 **Apply AI to real-world applications** like game AI, robotics, and autonomous agents.

📣 **Note:** This challenge offers a fun yet practical way to improve RL skills while building an intelligent, self-learning AI hunter!

