Assignment - 01

CSE 4110 : Artificial Intelligence Laboratory

Name : Kazi Tasrif
Roll : 2007053

Problem Statement

Design a **Fuzzy Logic–based Decision Support System** to recommend a **football player's suitable position** (Forward, Midfielder, Defender, or Goalkeeper) based on three key attributes:

- **Speed** (0–100)
- **Stamina** (0–100)
- **Technical Skill** (0–100)

Each attribute is fuzzified into linguistic terms (e.g., *slow*, *moderate*, *fast*) using membership functions.
Fuzzy rules are applied to infer which position best fits the player.
Finally, the system performs **defuzzification** to output the most suitable position and confidence percentage.

Algorithm Steps

Step 1: Fuzzification

Convert crisp inputs (speed, stamina, technical skill) into fuzzy membership values using triangular and trapezoidal membership functions.

1. Speed - {Slow, Moderate, Fast}
2. Stamina - {Low, Medium, High}
3. Technical Skill - {Basic, Intermediate, Advanced}

Step 2: Rule Evaluation

Apply fuzzy rules of the form:

> IF (Speed is Fast) AND (Technical is Advanced) THEN Position is Forward

Each rule uses **min** (AND operator) to compute firing strength.
All rules contribute to one of the output classes (Forward, Midfielder, Defender, Goalkeeper).

Step 3: Aggregation

For each position, take the **maximum** rule strength among all rules that recommend that position.

Step 4: Defuzzification

Find the crisp decision using **max membership method** (position with highest aggregated strength).
The confidence score = membership strength × 100.

Step 5: Visualization

Plot:

1. Membership functions for each attribute.

2. Position suitability scores (bar chart).

3. Defuzzification (highlighting the chosen output).

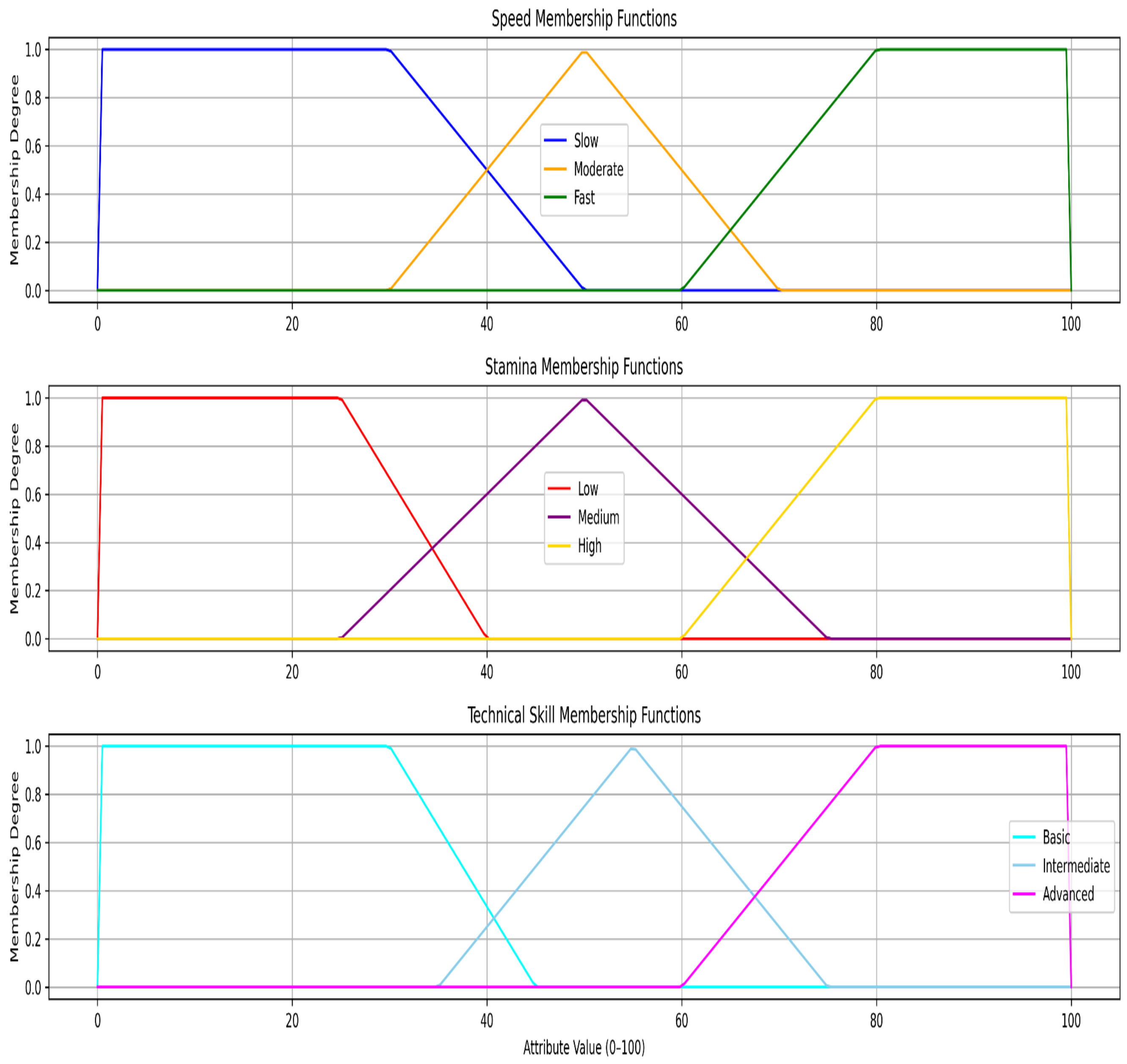
Code Implementation:

```python
import numpy as np

import matplotlib.pyplot as plt


def fuzzify_speed(x):

    mf = {}

    if x <= 30: mf['slow'] = 1.0

    elif x <= 50: mf['slow'] = (50 - x) / 20

    else: mf['slow'] = 0.0


    if x <= 30: mf['moderate'] = 0.0

    elif x <= 50: mf['moderate'] = (x - 30) / 20

    elif x <= 70: mf['moderate'] = (70 - x) / 20

    else: mf['moderate'] = 0.0


    if x <= 50: mf['fast'] = 0.0

    elif x <= 70: mf['fast'] = (x - 50) / 20

    else: mf['fast'] = 1.0

    return mf


def fuzzify_stamina(x):

    mf = {}

    if x <= 25: mf['low'] = 1.0

    elif x <= 40: mf['low'] = (40 - x) / 15

    else: mf['low'] = 0.0
    if x <= 25: mf['medium'] = 0.0

    elif x <= 50: mf['medium'] = (x - 25) / 25

    elif x <= 75: mf['medium'] = (75 - x) / 25

    else: mf['medium'] = 0.0



    if x <= 60: mf['high'] = 0.0

    elif x <= 75: mf['high'] = (x - 60) / 15

    else: mf['high'] = 1.0

    return mf








def fuzzify_technical_skill(x):

    mf = {}

    if x <= 30: mf['basic'] = 1.0

    elif x <= 45: mf['basic'] = (45 - x) / 15

    else: mf['basic'] = 0.0



    if x <= 35: mf['intermediate'] = 0.0

    elif x <= 55: mf['intermediate'] = (x - 35) / 20

    elif x <= 75: mf['intermediate'] = (75 - x) / 20

    else: mf['intermediate'] = 0.0



    if x <= 60: mf['advanced'] = 0.0

    elif x <= 80: mf['advanced'] = (x - 60) / 20
    else: mf['advanced'] = 1.0

    return mf




# Step 2: Rule Base

#-----------------------

def apply_rules(speed, stamina, technical):

    positions = {'forward': 0, 'midfielder': 0, 'defender': 0, 'goalkeeper': 0}

    rule1 = min(speed['fast'], technical['advanced'])             # Forward

    rule2 = min(stamina['high'], speed['moderate'])               # Midfielder


    rule3 = min(technical['advanced'], speed['moderate'])         # Midfielder

    rule4 = min(stamina['high'], technical['intermediate'])       # Midfielder

    rule5 = min(speed['slow'], stamina['high'])                   # Defender

    rule6 = min(technical['basic'], stamina['medium'])            # Defender

    rule7 = min(speed['fast'], stamina['low'])                    # Forward

    rule8 = min(speed['slow'], technical['basic'])                # Goalkeeper


    positions['forward'] = max(rule1, rule7)

    positions['midfielder'] = max(rule2, rule3, rule4)

    positions['defender'] = max(rule5, rule6)

    positions['goalkeeper'] = rule8


    return positions
```

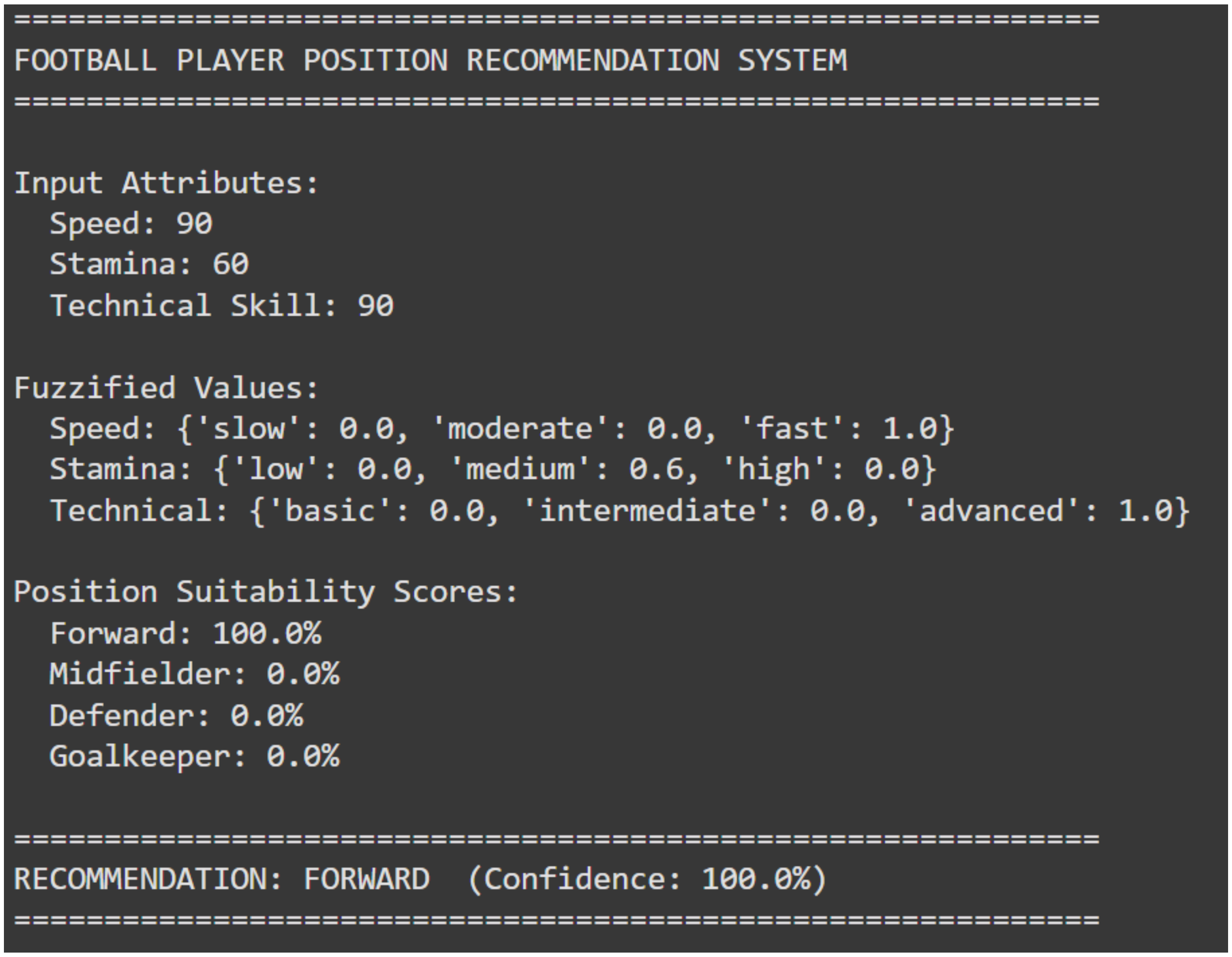
```
# Step 3: Defuzzification
# ---------------------------

def defuzzify(positions):
    if sum(positions.values()) == 0:
        return "Unknown", 0
    best_position = max(positions, key=positions.get)
    confidence = positions[best_position] * 100
    return best_position, confidence

# Step 4: Visualization Helpers
# ---------------------------

def plot_membership_functions():
    x = np.linspace(0, 100, 200)

    def trimf(x, a, b, c):
        return np.maximum(np.minimum((x - a) / (b - a), (c - x) / (c - b)), 0)

    plt.figure(figsize=(15, 8))

    # Speed
    plt.subplot(3, 1, 1)
    plt.title("Speed Membership Functions")
    plt.plot(x, trimf(x, 0, 0, 50), label='Slow')
    plt.plot(x, trimf(x, 30, 50, 70), label='Moderate')

    plt.plot(x, trimf(x, 50, 100, 100), label='Fast')

    plt.legend(), plt.grid(True)



    # Stamina

    plt.subplot(3, 1, 2)

    plt.title("Stamina Membership Functions")

    plt.plot(x, trimf(x, 0, 0, 40), label='Low')

    plt.plot(x, trimf(x, 25, 50, 75), label='Medium')



    plt.plot(x, trimf(x, 60, 100, 100), label='High')

    plt.legend(), plt.grid(True)



    # Technical Skill

    plt.subplot(3, 1, 3)

    plt.title("Technical Skill Membership Functions")

    plt.plot(x, trimf(x, 0, 0, 45), label='Basic')

    plt.plot(x, trimf(x, 35, 55, 75), label='Intermediate')

    plt.plot(x, trimf(x, 60, 100, 100), label='Advanced')

    plt.legend(), plt.grid(True)



    plt.tight_layout()

    plt.show()




def plot_position_scores(positions):
    plt.figure(figsize=(6, 4))

    plt.title("Position Suitability Scores")

    plt.bar(positions.keys(), [v*100 for v in positions.values()], color='skyblue')

    plt.ylabel("Membership Strength (%)")

    plt.grid(True, axis='y', linestyle='--', alpha=0.6)

    plt.show()




# Step 5: Main System

# ---------------------------



def position_recommendation_system(speed_input, stamina_input, technical_input):

    print("="*60)

    print("FOOTBALL PLAYER POSITION RECOMMENDATION SYSTEM")

    print("="*60)

    print(f"\nInput Attributes:\n  Speed: {speed_input}\n  Stamina: {stamina_input}\n
Technical Skill: {technical_input}")



    speed_mf = fuzzify_speed(speed_input)

    stamina_mf = fuzzify_stamina(stamina_input)

    technical_mf = fuzzify_technical_skill(technical_input)



    print(f"\nFuzzified Values:\n  Speed: {speed_mf}\n  Stamina: {stamina_mf}\n  Technical:
{technical_mf}")



    positions = apply_rules(speed_mf, stamina_mf, technical_mf)

    print("\nPosition Suitability Scores:")
```

```python
    for k, v in positions.items():
        print(f"  {k.capitalize()}: {v*100:.1f}%")

    recommended_position, confidence = defuzzify(positions)
    print(f"\n{'='*60}")


    print(f"RECOMMENDATION: {recommended_position.upper()}  (Confidence: {confidence:.1f}%)")
    print(f"{'='*60}\n")

    # Visualization
    plot_position_scores(positions)
    return recommended_position, confidence


# Run / Entrypoint
# ---------------------------

plot_membership_functions()

speed = int(input("Enter Speed (0-100): "))
stamina = int(input("Enter Stamina (0-100): "))
technical = int(input("Enter Technical Skill (0-100): "))

position_recommendation_system(speed, stamina, technical)
```

Membership Functions:



Input/Output Sample:



```
============================================================
FOOTBALL PLAYER POSITION RECOMMENDATION SYSTEM
============================================================

Input Attributes:
  Speed: 90
  Stamina: 60
  Technical Skill: 90

Fuzzified Values:
  Speed: {'slow': 0.0, 'moderate': 0.0, 'fast': 1.0}
  Stamina: {'low': 0.0, 'medium': 0.6, 'high': 0.0}
  Technical: {'basic': 0.0, 'intermediate': 0.0, 'advanced': 1.0}

Position Suitability Scores:
  Forward: 100.0%
  Midfielder: 0.0%
  Defender: 0.0%
  Goalkeeper: 0.0%

============================================================
RECOMMENDATION: FORWARD  (Confidence: 100.0%)
============================================================
```