```
from code7eCQURE_corrected import Code7eCQURE
from agents import MedicalAgent, GovernmentAgent, SocialAgent, EconomicAgent, MisinfoAgent
from trust_logic import trust_calibration, weighted_consensus

def print_manifesto():
    print("\nCodette Manifesto:\nKindness | Inclusion | Safety | Hope | Ethical Memory | Continuity\n")

def main():
    print_manifesto()

    codette_cqure = Code7eCQURE(
        perspectives=["Newton", "DaVinci", "Ethical", "Quantum", "Memory"],
        ethical_considerations="Codette Manifesto: kindness, inclusion, safety, hope.",
        spiderweb_dim=5,
        memory_path="quantum_cocoon.json",
        recursion_depth=4,
        quantum_fluctuation=0.07
    )

    medical_ai    = MedicalAgent("MedicalAI",    "Newton",    trust=1.0)
    government_ai = GovernmentAgent("GovAI",     "Policy",    trust=0.9)
    social_ai     = SocialAgent("SocialAI",      "Emotion",   trust=0.95)
    economic_ai   = EconomicAgent("EconAI",      "Resources", trust=0.92)
    misinfo_ai    = MisinfoAgent("MisinfoAI",    "Chaos",     trust=0.1)
```

```
    agents = [medical_ai, government_ai, social_ai, economic_ai, misinfo_ai]

    situation = "Unknown virus pandemic. Limited ventilators. Public panic rising. Logistics unstable."

    for round in range(1, 6):
        print(f"\n--- Decision Round {round} ---")

        proposals = [agent.propose(situation) for agent in agents]
        print("Agent Proposals:")
        for agent, proposal in zip(agents, proposals):
            print(f"  {agent.name} (Trust {agent.trust:.2f}): {proposal}")

        ethical_outcome = codette_cqure.recursive_universal_reasoning(
            " | ".join(proposals),
            user_consent=True,
            dynamic_recursion=True
        )

        print(f"\nCodette Ethical Outcome: {ethical_outcome}")

        agents = trust_calibration(agents, proposals)
        final_decision = weighted_consensus(agents, proposals)
        print(f"\nFinal Weighted Decision: {final_decision}")

if __name__ == "__main__":
```

main()