```
import numpy as np

class ZomatoOrderNeuron:
    def __init__(self):
        # Define weights for each feature (hunger, distance, affordability)
        self.weights = np.array([0.5, 0.3, 0.2])
        self.threshold = 0.5

    def decide_order(self, features):

        # Calculate weighted sum
        weighted_sum = np.dot(features, self.weights)

        # Apply threshold function
        return 1 if weighted_sum >= self.threshold else 0

# Example usage
if __name__ == "__main__":
    # Initialize the neuron
    order_neuron = ZomatoOrderNeuron()

    # Test different scenarios
    test_cases = [
        [1, 1, 1],  # Hungry, close, affordable - ORDER
        [1, 0, 0],  # Hungry but far and expensive - ORDER (due to hunger)
        [0, 1, 1],  # Not hungry but close and affordable - ORDER
        [0, 0, 1],  # Not hungry, far, but affordable - NO ORDER
        [0, 0, 0],  # Not hungry, far, expensive - NO ORDER
        [1, 1, 0],  # Hungry, close, but expensive - ORDER
    ]

    print("Zomato Order Decision System")
    print("Features: [Hunger, Distance, Affordability]")
    print("Threshold:", order_neuron.threshold)
    print("Weights:", order_neuron.weights)
    print("\nTest Cases:")

    for i, features in enumerate(test_cases, 1):
        decision = order_neuron.decide_order(features)
        print(f"Case {i}: {features} -> Order: {decision} ({'PLACE ORDER' if decision == 1 else 'NO ORDER'})")
```

```
Zomato Order Decision System
Features: [Hunger, Distance, Affordability]
Threshold: 0.5
Weights: [0.5 0.3 0.2]

Test Cases:
Case 1: [1, 1, 1] -> Order: 1 (PLACE ORDER)
Case 2: [1, 0, 0] -> Order: 1 (PLACE ORDER)
Case 3: [0, 1, 1] -> Order: 1 (PLACE ORDER)
Case 4: [0, 0, 1] -> Order: 0 (NO ORDER)
Case 5: [0, 0, 0] -> Order: 0 (NO ORDER)
Case 6: [1, 1, 0] -> Order: 1 (PLACE ORDER)
```