ASTRO CALCULATIONS

The process outlined relies on a set of numerological algorithms combined with astrological and industry-specific principles to analyze and calculate compatibility. Although these methods aren't "algorithms" in the traditional computer science sense, they do follow a defined set of rules and logical steps. Below are the key types of algorithms or rule-based calculations used in this process:

1. Primary Number Calculation (PN) Algorithm:

- Input: Day of birth (numerical value)
- Process: Add the digits of the day of birth together. If the result is a two-digit number, reduce it by summing its digits until a single digit is achieved.
- Example Algorithm:

```python
def calculate_primary_number(day):
    while day >= 10:
        day = sum(map(int, str(day)))
    return day
```

- Use: This algorithm derives the Primary Number (PN) from the birth date.

2. Destiny Number Calculation (DN) Algorithm:

- Input: Full date of birth (day, month, year)
- Process: Add the digits of the day, month, and year of birth together. Reduce the total sum to a single digit.
- Example Algorithm:

```python
def calculate_destiny_number(day, month, year):
    total = sum(map(int, str(day))) + sum(map(int, str(month))) + sum(map(int, str(year)))
    while total >= 10:
        total = sum(map(int, str(total)))
    return total
```

- Use: This algorithm calculates the Destiny Number (DN) by summing and reducing the entire birth date to a single digit.

3. Kua Number (KN) Algorithm:

- Input: Year of birth and gender
- Process:
    - For males, sum the last two digits of the birth year and subtract the result from 10.
    - For females, sum the last two digits of the birth year and add 5 to the result.
- Reduce the result to a single digit.
- Example Algorithm:

```python
def calculate_kua_number(year, gender):
    last_two_digits = int(str(year)[-2:])
    kua = sum(map(int, str(last_two_digits)))
    if gender == 'male':
        kua = 10 - kua
    else:
        kua += 5
    while kua >= 10:
        kua = sum(map(int, str(kua)))
    return kua
```

- Use: This algorithm calculates the Kua Number (KN) based on gender and birth year.

4. Common Compatible Numbers (CCPN and CCDN) Algorithm:

- Input: PN and DN of multiple owners
- Process: Determine the compatibility between multiple owners' Primary and Destiny Numbers using the Name Number Compatibility Table.
- Example Approach:

```python
def is_compatible(number1, number2, compatibility_table):
    # Check compatibility between number1 and number2 based on the given table
```

```
    if number2 in compatibility_table['friends'][number1]:
        return 'friends'
    elif number2 in compatibility_table['neutral'][number1]:
        return 'neutral'
    else:
        return 'enemy'
```

- Use: This algorithm checks if the Primary Numbers (PNs) or Destiny Numbers (DNs) of multiple owners are compatible by looking up friendly, neutral, or enemy relationships.

5. Karaka Planet Number (KPN) Assignment Algorithm:

- Input: Industry type
- Process: Map the industry type to a specific set of planetary numbers (KPNs) based on predefined rules.
- Example Algorithm:

```python
industry_kpn = {
    'Agriculture': [3, 6],
    'Healthcare': [3, 5],
    'IT': [4, 5],
    'Real Estate': [8, 9],
    # Add more industries and KPN mappings here...
}
def get_kpn_for_industry(industry):
    return industry_kpn.get(industry, None)
```

- Use: This algorithm maps the industry to the relevant Karaka Planet Numbers (KPN).

6. Name Number Calculation (NN) Algorithm:

- Input: Proposed company name
- Process:
    - Assign a numerical value to each letter of the company name based on the numerology chart.
    - Sum the values of all letters in the company name and reduce the result to a single digit.
- Example Algorithm:

```python
letter_values = {
    'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 8, 'G': 3, 'H': 5, 'I': 1, 'J': 1,
    'K': 2, 'L': 3, 'M': 4, 'N': 5, 'O': 7, 'P': 8, 'Q': 1, 'R': 2, 'S': 3, 'T': 4,
    'U': 6, 'V': 6, 'W': 6, 'X': 5, 'Y': 1, 'Z': 7
}
def calculate_name_number(name):
    total = sum(letter_values[char.upper()] for char in name if char.isalpha())
    while total >= 10:
        total = sum(map(int, str(total)))
    return total
```

- Use: This algorithm calculates the Name Number (NN) of the proposed company name.

7. Compatibility Check Algorithm:

- Input: NN, KPN, CCPN, CCDN, and DN of the company
- Process:
    - Compare the Name Number (NN) with the KPN, CCPN, CCDN, and DN based on the Name Number Compatibility Table.
    - Determine whether the name is "favorable," "neutral," or "unfavorable" based on the relationships between these numbers.
- Example Approach:

```python
def check_name_compatibility(nn, kpn, ccpn, dn, compatibility_table):
```

```
    kpn_check = is_compatible(nn, kpn, compatibility_table)
    ccpn_check = is_compatible(nn, ccpn, compatibility_table)
    dn_check = is_compatible(nn, dn, compatibility_table)
    # Combine results to determine rating
    if kpn_check == 'friends' and ccpn_check == 'friends' and dn_check == 'friends':
        return 'Most Favourable'
    elif (kpn_check == 'friends' and ccpn_check == 'neutral') or (ccpn_check == 'friends' and dn_check == 'neutral'):
        return 'Favourable'
    else:
        return 'Neutral or Unfavourable'
```

- Use: This algorithm rates the compatibility of the Name Number (NN) with the other numerological values.

Summary:

These steps involve a combination of arithmetic reduction algorithms, lookup tables, and logical condition checks to evaluate compatibility between various numerological and astrological factors. The entire process mimics a decision tree structure, where the most favorable name is selected based on how well the Name Number aligns with the personal numbers (PN, DN, KN) of the owners and the industry's planetary influences (KPN).