9 Star Ki Calculator Algorithms and Pseudocode

Version 1.0 - Implementation-Ready Specification

---

1. Core Algorithm: Principal Star Calculation

Algorithm: `computePrincipalStar(date, time, timezone, method='traditional')`

Purpose: Calculate the year star (Honmei/æœ¬å‘½æ˜Ÿ) from birth date.

Inputs:
- `date`: Birth date (YYYY-MM-DD)
- `time`: Birth time (HH:MM:SS, optional for year calculation)
- `timezone`: IANA timezone identifier (e.g., 'America/New_York')
- `method`: 'traditional' or 'chinese'

Outputs:
- Integer 1-9 representing the principal star

Pseudocode:

```
FUNCTION computePrincipalStar(date, time, timezone, method):
    // Step 1: Convert to local datetime
    local_dt = convert_to_local_time(date, time, timezone)

    // Step 2: Determine solar year
    solar_year = determineSolarYear(local_dt, timezone)

    // Step 3: Calculate principal star using traditional method
    IF method == 'traditional' OR (method == 'chinese' AND gender == 'male'):
        // Sum all digits of the year
        digit_sum = sumDigits(solar_year)

        // Reduce to single digit
        WHILE digit_sum >= 10:
            digit_sum = sumDigits(digit_sum)

        // Apply formula: 11 - digit_sum
        principal_star = 11 - digit_sum

        // If result is 10, reduce again
```

```
        IF principal_star >= 10:
            principal_star = sumDigits(principal_star)

    ELSE IF method == 'chinese' AND gender == 'female':
        // Use ascending (Yin) spiral
        // Formula: 2 + digit_sum
        digit_sum = sumDigits(solar_year)
        WHILE digit_sum >= 10:
            digit_sum = sumDigits(digit_sum)

        principal_star = (2 + digit_sum)
        IF principal_star > 9:
            principal_star = principal_star - 9

    RETURN principal_star

FUNCTION sumDigits(number):
    sum = 0
    FOR EACH digit IN str(number):
        sum = sum + int(digit)
    RETURN sum

FUNCTION determineSolarYear(local_dt, timezone):
    year = local_dt.year

    // Get Li Chun moment for this year
    li_chun_moment = getLiChunMoment(year, timezone)

    // If birth is before Li Chun, use previous year
    IF local_dt < li_chun_moment:
        RETURN year - 1
    ELSE:
        RETURN year

FUNCTION getLiChunMoment(year, timezone):
    // Li Chun occurs when sun reaches 315Â° celestial longitude
    // Typical dates: Feb 3-5, around 22:00-02:00 UTC

    // OPTION 1: Use lookup table for specific years
    IF year IN li_chun_table:
        utc_moment = li_chun_table[year]
        RETURN convert_to_timezone(utc_moment, timezone)

    // OPTION 2: Use astronomical calculation
```

```
   // (requires ephemeris library)
   utc_moment = calculate_solar_longitude_moment(year, 315.0)
   RETURN convert_to_timezone(utc_moment, timezone)

   // OPTION 3: Use default approximation
   // WARNING: May be off by 1-2 days in edge cases
   RETURN datetime(year, 2, 4, 0, 0, 0, timezone)
```

Boundary Case Handling:

```
TEST CASE: Birth on Feb 3, 2024, 11:00 PM EST
   - Li Chun 2024 UTC: Feb 4, 2024, 04:28 UTC
   - Local time: Feb 3, 2024, 23:00 EST (Feb 4, 04:00 UTC)
   - Birth is BEFORE Li Chun (04:28 UTC)
   - Use 2023 for calculation
   - Expected: Principal star for 2023

TEST CASE: Birth on Feb 4, 2024, 1:00 AM EST
   - Li Chun 2024 UTC: Feb 4, 2024, 04:28 UTC
   - Local time: Feb 4, 2024, 01:00 EST (Feb 4, 06:00 UTC)
   - Birth is AFTER Li Chun
   - Use 2024 for calculation
   - Expected: Principal star for 2024
```

---

2. Core Algorithm: Month Star Calculation

Algorithm: `computeMonthStar(date, time, timezone, principal_star, method='traditional')`

Purpose: Calculate the month star (Getsumei/æœˆå‘½æ˜Ÿ) from birth date.

Inputs:
- `date`, `time`, `timezone`: Birth datetime information
- `principal_star`: Previously calculated principal star (1-9)
- `method`: Calculation method

Outputs:
- Integer 1-9 representing the month star

Pseudocode:

```
FUNCTION computeMonthStar(date, time, timezone, principal_star, method):
    // Step 1: Convert to local datetime
    local_dt = convert_to_local_time(date, time, timezone)

    // Step 2: Determine which solar month
    solar_month = determineSolarMonth(local_dt, timezone)

    // Step 3: Determine principal star group
    IF principal_star IN [1, 4, 7]:
        group = "147"
    ELSE IF principal_star IN [2, 5, 8]:
        group = "258"
    ELSE IF principal_star IN [3, 6, 9]:
        group = "369"

    // Step 4: Look up month star from table
    month_star = MONTH_STAR_TABLE[group][solar_month]

    RETURN month_star

// Month star lookup table
MONTH_STAR_TABLE = {
    "147": {
        "February": 8, "March": 7, "April": 6, "May": 5,
        "June": 4, "July": 3, "August": 2, "September": 1,
        "October": 9, "November": 8, "December": 7, "January": 6
    },
    "258": {
        "February": 2, "March": 1, "April": 9, "May": 8,
        "June": 7, "July": 6, "August": 5, "September": 4,
        "October": 3, "November": 2, "December": 1, "January": 9
    },
    "369": {
        "February": 5, "March": 4, "April": 3, "May": 2,
        "June": 1, "July": 9, "August": 8, "September": 7,
        "October": 6, "November": 5, "December": 4, "January": 3
    }
}

FUNCTION determineSolarMonth(local_dt, timezone):
    // Get solar term start dates for the year
    year = local_dt.year
```

```
    solar_terms = getSolarTermsForYear(year, timezone)

    // Find which solar month by comparing to solar term boundaries
    // Solar months begin at: Li Chun, Jing Zhe, Qing Ming, Li Xia, etc.

    month_boundaries = [
        ("January", solar_terms["Xiao Han"]),      // ~Jan 5-6
        ("February", solar_terms["Li Chun"]),      // ~Feb 3-5
        ("March", solar_terms["Jing Zhe"]),        // ~Mar 5-6
        ("April", solar_terms["Qing Ming"]),       // ~Apr 4-5
        ("May", solar_terms["Li Xia"]),            // ~May 5-6
        ("June", solar_terms["Mang Zhong"]),       // ~Jun 5-6
        ("July", solar_terms["Xiao Shu"]),         // ~Jul 6-7
        ("August", solar_terms["Li Qiu"]),         // ~Aug 7-8
        ("September", solar_terms["Bai Lu"]),      // ~Sep 7-8
        ("October", solar_terms["Han Lu"]),        // ~Oct 8-9
        ("November", solar_terms["Li Dong"]),      // ~Nov 7-8
        ("December", solar_terms["Da Xue"])        // ~Dec 6-7
    ]

    // Find the current solar month
    FOR i FROM len(month_boundaries) - 1 DOWN TO 0:
        month_name, boundary_dt = month_boundaries[i]
        IF local_dt >= boundary_dt:
            RETURN month_name

    // Default: should not reach here if solar terms are complete
    RETURN "January"

FUNCTION getSolarTermsForYear(year, timezone):
    // OPTION 1: Use pre-calculated table
    IF year IN SOLAR_TERMS_TABLE:
        RETURN SOLAR_TERMS_TABLE[year]

    // OPTION 2: Calculate astronomically
    solar_terms = {}
    solar_terms["Li Chun"] = calculate_solar_longitude_moment(year, 315.0)
    solar_terms["Jing Zhe"] = calculate_solar_longitude_moment(year, 345.0)
    // ... etc for all 24 solar terms

    // Convert all to local timezone
    FOR term IN solar_terms:
        solar_terms[term] = convert_to_timezone(solar_terms[term], timezone)
```

    RETURN solar_terms

    // OPTION 3: Use approximation (less accurate)
    RETURN approximate_solar_terms(year, timezone)
```

### Month Boundary Example:

```
TEST CASE: Birth on March 5, 2024, 10:00 AM UTC
    - Jing Zhe 2024: March 5, 2024, 10:22 UTC
    - Birth is BEFORE Jing Zhe boundary
    - Should use FEBRUARY solar month
    - Principal star: assume 5 (group 258)
    - Month star: MONTH_STAR_TABLE["258"]["February"] = 2

TEST CASE: Birth on March 5, 2024, 11:00 AM UTC
    - Jing Zhe 2024: March 5, 2024, 10:22 UTC
    - Birth is AFTER Jing Zhe boundary
    - Should use MARCH solar month
    - Principal star: assume 5 (group 258)
    - Month star: MONTH_STAR_TABLE["258"]["March"] = 1
```

---

3. Core Algorithm: Energetic/Superficial Star

Algorithm: `computeEnergeticStar(principal_star, month_star, method='japanese')`

Purpose: Calculate the third star (energetic/superficial/tendency star).

Inputs:
- `principal_star`: Previously calculated principal star (1-9)
- `month_star`: Previously calculated month star (1-9)
- `method`: 'japanese' (Lo Shu based) or 'chinese' (day star based)

Outputs:
- Integer 1-9 representing the energetic star

Pseudocode (Japanese/Lo Shu Method):

```
FUNCTION computeEnergeticStar_Japanese(principal_star, month_star):

```
    // Use Lo Shu combination lookup
    // The energetic star is determined by the position of the month star
    // relative to the principal star in the Lo Shu square

    // Pre-computed 81 combination table (9 Ã— 9 matrix)
    energetic_star = ENERGETIC_STAR_TABLE[principal_star][month_star]

    RETURN energetic_star

// 81 Combination Table (partial - full table required for implementation)
// Each row is principal star, each column is month star
ENERGETIC_STAR_TABLE = {
    1: {1: 1, 2: 2, 3: 3, 4: 4, 5: 5, 6: 6, 7: 7, 8: 8, 9: 9},
    2: {1: 2, 2: 2, 3: 8, 4: 4, 5: 2, 6: 6, 7: 4, 8: 8, 9: 1},
    // ... (full 81 combinations needed)
    // Note: This table must be validated against Japanese sources
}

ALTERNATIVE FORMULA (Lo Shu position-based):
FUNCTION computeEnergeticStar_LoShu(principal_star, month_star):
    // Lo Shu positions
    lo_shu_positions = {
        1: (1, 0), 2: (2, 0), 3: (0, 1),
        4: (0, 0), 5: (1, 1), 6: (2, 2),
        7: (2, 1), 8: (0, 2), 9: (1, 2)
    }

    // Get positions
    prin_pos = lo_shu_positions[principal_star]
    month_pos = lo_shu_positions[month_star]

    // Calculate offset
    offset_row = month_pos[0] - prin_pos[0]
    offset_col = month_pos[1] - prin_pos[1]

    // Apply offset to Lo Shu center (5) to find energetic star
    // This is a simplified version - actual implementation may vary
    energetic_pos = (1 + offset_row, 1 + offset_col)

    // Look up which star is at this position
    FOR star, position IN lo_shu_positions:
        IF position == energetic_pos:
            RETURN star
```

    RETURN 5 // Default to center
```

**Note**: The exact formula for the Japanese method's third star is not fully standardized in sources. Implementation should:
1. Use a validated 81-combination lookup table from a reliable Japanese or Western practitioner source
2. OR provide a toggle to let users choose whether to include the third star
3. Cite the specific source used for the combination table

---

## 4. Complete Calculator Function

### Algorithm: `calculate9StarKi(birth_date, birth_time, timezone, gender=None, method='traditional')`

**Complete Implementation**:

```
FUNCTION calculate9StarKi(birth_date, birth_time, timezone, gender, method):
    // Input validation
    VALIDATE_DATE(birth_date)
    VALIDATE_TIME(birth_time)
    VALIDATE_TIMEZONE(timezone)

    IF method == 'chinese' AND gender IS NULL:
        THROW ERROR("Gender required for Chinese method")

    // Step 1: Calculate principal star
    principal_star = computePrincipalStar(
        birth_date, birth_time, timezone, method
)

    // Step 2: Calculate month star
    month_star = computeMonthStar(
        birth_date, birth_time, timezone, principal_star, method
)

    // Step 3: Calculate energetic star (optional)
    energetic_star = computeEnergeticStar_Japanese(
        principal_star, month_star
)

```
    // Step 4: Get element metadata
    principal_element = ELEMENT_METADATA[principal_star]
    month_element = ELEMENT_METADATA[month_star]
    energetic_element = ELEMENT_METADATA[energetic_star]

    // Return complete profile
    RETURN {
        "principal_star": principal_star,
        "month_star": month_star,
        "energetic_star": energetic_star,
        "principal_element": principal_element,
        "month_element": month_element,
        "energetic_element": energetic_element,
        "method_used": method,
        "solar_year_used": determineSolarYear(local_dt, timezone),
        "solar_month_used": determineSolarMonth(local_dt, timezone),
        "boundary_notes": checkBoundaryWarnings(birth_date, birth_time, timezone)
    }

FUNCTION checkBoundaryWarnings(birth_date, birth_time, timezone):
    warnings = []
    local_dt = convert_to_local_time(birth_date, birth_time, timezone)

    // Check if near Li Chun
    year = local_dt.year
    li_chun = getLiChunMoment(year, timezone)
    time_diff = abs(local_dt - li_chun)

    IF time_diff < 2 DAYS:
        warnings.append(
            "Birth is within 2 days of Li Chun boundary. " +
            "Solar year determination is critical."
        )

    // Check if near month boundary
    solar_terms = getSolarTermsForYear(year, timezone)
    FOR term_name, term_moment IN solar_terms:
        time_diff = abs(local_dt - term_moment)
        IF time_diff < 1 DAY:
            warnings.append(
                f"Birth is within 1 day of {term_name} solar term. " +
                "Solar month determination should be verified."
            )
```

    RETURN warnings
```

---

## 5. Implementation Notes

### Required Data Tables:

1. **Li Chun Moments Table** (2000-2050 recommended minimum)
   - Format: `{year: UTC_datetime}`
   - Source: Astronomical calculation or perpetual calendar

2. **Solar Terms Table** (all 24 terms)
   - Format: `{year: {term_name: UTC_datetime}}`
   - Update annually or calculate astronomically

3. **81 Combinations Table** (for energetic star)
   - Format: 9x9 matrix
   - Source: Must cite specific Japanese or Western 9 Ki source

### Error Handling:

```
- Invalid date/time â†’ Clear error message
- Missing timezone â†’ Default to UTC with warning
- Birth before year 1900 â†’ Warning about calculation reliability
- Birth after year 2100 â†’ Warning about solar term table coverage
- Method = 'chinese' without gender â†’ Error
```

Performance Considerations:

- Cache solar term calculations
- Pre-compute and store Li Chun moments for common years
- Optimize timezone conversions

---

6. Testing Strategy

Unit Tests Required:

1. **Principal star calculation**
```

   - Test years: 1950, 1965, 1972, 1980, 1986, 1990, 1995, 1999, 2000, 2020
   - Boundary dates: Feb 3, Feb 4, Feb 5 for multiple years
   - Cross-century transitions

2. **Month star calculation**
   - All 12 solar months
   - Month boundary dates (within 24 hours of solar term)
   - All 3 principal star groups (1-4-7, 2-5-8, 3-6-9)

3. **Timezone handling**
   - UTC, EST, PST, JST, GMT
   - Births at midnight
   - Births during DST transitions

4. **Method comparison**
   - Same birth date, both methods
   - Verify gender differences in Chinese method

Integration Tests:

1. Complete profiles for known individuals
2. Cross-validation with established 9 Ki calculators
3. Boundary case scenarios (Feb 3-5, near month transitions)

Validation Sources:

- Compare against: 9starkiastrology.com calculator
- Compare against: heluo.nl day star calendar
- Verify with Japanese almanac (ä¸‡å¹´æš¦)

Appendix A: Solar Term Calculation

For precise implementation, use astronomical library (e.g., PyEphem, Skyfield) to calculate exact moments when sun reaches specific celestial longitudes:

```
Li Chun (ç«‹æ˜¥): 315Â°
Jing Zhe (æƒŠèœ‡): 345Â°
Qing Ming (æ¸…æ˜Ž): 15Â°
... (all 24 terms at 15Â° intervals)
```

Alternative: Use published perpetual calendar (ä¸‡å¹´æš¦) data.

End of Algorithm Specification
Version 1.0 | 2025-10-30