1. Problem Analysis

Step 1: Identify Requirements

- Look for patterns in what's being asked:
  - Domain control (Wk(n))
  - Range control (Ek(n))
  - Both domain and range control
  - Size control (|Wk(n)|)

Step 2: Target Properties

Write down clearly:

```
- What should be in Wk(n)?
- What should be in Ek(n)?
- Any special conditions (totality, size)?
```

2. Solution Construction

Step 1: Helper Function Design

```
# Pattern for domain control:
f(n,x) = { something  if x in desired_domain
         { ↑          otherwise

# Pattern for range control:
f(n,x) = { desired_output  if condition
         { default_value   otherwise

# Pattern for both:
f(n,x) = { desired_output  if x in desired_domain
         { ↑              otherwise
```

Step 2: Make it Computable

Common techniques:

```
- For divisibility: use rm(x,n)
- For domain control: use μz
- For even numbers: multiply by 2
- For size control: use bounded counters
```

Step 3: Apply SMN Theorem

Standard steps:

1. State the theorem
2. Show helper function is computable
3. Get k(n) such that φk(n)(x) = f(n,x)

3. Verification

Step 1: Check Domain (Wk(n))

```
Wk(n) = {x | f(n,x)↓}
Verify this matches requirements
```

Step 2: Check Range (Ek(n))

```
Ek(n) = {f(n,x) | x ∈ Wk(n)}
Verify this matches requirements
```

Step 3: Verify Special Properties

- Check totality if required
- Verify size conditions
- Confirm any other constraints

4. Common Patterns

Domain Control

```
# For x ≥ n:
f(n,x) = something + μz.(n-x)
```

```
# For x < n:
f(n,x) = something + μz.(x-n)
```

Range Control

```
# For even numbers:
f(n,x) = 2*something

# For specific sets:
f(n,x) = desired_value * sg(condition) + default * sg(condition)
```

Size Control

```
# For fixed size k:
f(n,x) = { 0  if x < k
         { ↑  otherwise
```

5. Tips and Tricks

1. For totality:
   - Include a default value
   - Make sure every case is covered
2. For range control:
   - Use multiplication for even/odd
   - Use sign functions (sg, s$\overline{\text{g}}$) for switching
3. For domain control:
   - Use minimalization (μ)
   - Use remainder function (rm)
4. For verification:
   - Write out Wk(n) explicitly
   - Write out Ek(n) as set builder notation
   - Show transformations step by step

Remember: The key is matching the helper function design to the requirements while ensuring computability.