```
function find_invertible_happy_sequences()
    N = input('Enter the value for N for the transformation: ');
    M = input('Enter the number of transformation operations (M): ');

    % Generate forbidden subsequences
    forbidden_subsequences = generate_forbidden_subsequences(N);

    % Create initial sequences that are guaranteed to be invertible
    seq1 = [1, zeros(1, N)];
    seq2 = [1, zeros(1, N-1)];

    % Initialize variables
    happy_sequences = [];
    tested_sequences = {};
    unique_sequences = {};

    % Generate and test sequences
    for trial = 1:1000
        while true
            initial_seq = generate_random_sequence(seq1, seq2, 200,
forbidden_subsequences);
            if isempty(initial_seq)
                continue;
            end
            seq_str = num2str(initial_seq);
            if ~ismember(seq_str, tested_sequences)
                break;
            end
        end

        tested_sequences{end+1} = seq_str;
        seq = initial_seq;
        original_seq = seq;

        for j = 1:M
            seq = apply_forward_transformation(seq, N);
            new_seq = apply_backward_transformation(seq, N);

            if isempty(new_seq) || ~is_sequence_invertible(new_seq, N,
forbidden_subsequences)
                break;
            end

            seq = new_seq;

            if isequal(seq, original_seq)
                normalized_seq = sort(original_seq);
                if ~ismember(num2str(normalized_seq), unique_sequences)
                    unique_sequences{end+1} = num2str(normalized_seq);
                    happy_sequences = [happy_sequences; original_seq];
                end
```

```
                break;
            end
        end
    end

    disp('Unique happy sequences that regenerate themselves:');
    disp(happy_sequences);
end

function transformed = apply_forward_transformation(seq, N)
    transformed = [];
    for i = 1:length(seq)
        if seq(i) == 1
            transformed = [transformed, 1, zeros(1, N-1)];
        else
            transformed = [transformed, 1, zeros(1, N)];
        end
    end
end

function transformed = apply_backward_transformation(seq, N)
    transformed = [];
    i = 1;
    while i <= length(seq)
        if seq(i) == 1
            if (i + N) <= length(seq) && all(seq(i+1:i+N) == 0)
                transformed = [transformed, 0];
                i = i + N + 1;
            elseif (i + N - 1) <= length(seq) && all(seq(i+1:i+N-1) == 0)
                transformed = [transformed, 1];
                i = i + N;
            else
                return;
            end
        else
            return;
        end
    end
end

function initial_seq = generate_random_sequence(seq1, seq2, target_length,
forbidden_subsequences)
    initial_seq = [];
    while length(initial_seq) < target_length
        chosen_seq = randi([1, 2]);
        if chosen_seq == 1
            initial_seq = [initial_seq, seq1];
        else
            initial_seq = [initial_seq, seq2];
        end
        if any(contains(num2str(initial_seq), forbidden_subsequences))
            initial_seq = [];
            break;
        end
```

```
    end
    if ~isempty(initial_seq)
        initial_seq = initial_seq(1:target_length);
    end
end

function is_invertible = is_sequence_invertible(seq, N,
forbidden_subsequences)
    for forbidden_seq = forbidden_subsequences
        if contains(num2str(seq), forbidden_seq)
            is_invertible = false;
            return;
        end
    end
    is_invertible = true;
end

function forbidden_subsequences = generate_forbidden_subsequences(N)
    max_length = 2 * N;
    forbidden_subsequences = {};
    for len = 2:max_length
        for num = 0:(2^len - 1)
            subseq = de2bi(num, len);
            if ~is_subseq_invertible(subseq, N)
                forbidden_subsequences{end+1} = num2str(subseq);
            end
        end
    end
end

function is_invertible = is_subseq_invertible(subseq, N)
    cnt_ones = sum(subseq == 1);
    cnt_zeros = sum(subseq == 0);
    if cnt_ones >= 2 || cnt_zeros < N
        is_invertible = false;
    else
        is_invertible = true;
    end
end
```

```
Error using input
Cannot call INPUT from EVALC.

Error in find_invertible_happy_sequences (line 2)
    N = input('Enter the value for N for the transformation: ');
```

Published with MATLAB® R2023b