```
function shanon_fano_fn()
    % Define symbols and given probabilities
    symbols = ['A', 'B', 'C', 'D', 'E'];
    probabilities = [0.5, 0.2, 0.2, 0.05, 0.05];

    % Step 1: Sort symbols by probability (descending)
    [probabilities, idx] = sort(probabilities, 'descend');
    symbols = symbols(idx);

    % Step 2: Generate Shannon-Fano codes
    codes = cell(size(symbols));  % Initialize empty cell array for codes
    codes = shannon_encoder(1, length(symbols), probabilities, codes, '');

    % Display the Shannon-Fano codes
    disp('Symbol  Probability  Code');
    for i = 1:length(symbols)
        fprintf('%c       %.2f        %s\n', symbols(i), probabilities(i),
codes{i});
    end
end

% Recursive Shannon-Fano Encoding Function
function codes = shannon_encoder(begin_point, end_point, p, codes, prefix)
    if begin_point == end_point
        codes{begin_point} = prefix;
        return;
    end

    split_index = find_split(p(begin_point:end_point)) + begin_point - 1;

    codes = shannon_encoder(begin_point, split_index, p, codes, [prefix,
'0']);
    codes = shannon_encoder(split_index+1, end_point, p, codes, [prefix,
'1']);
end

% Find the optimal split point to balance probability sums
function split_index = find_split(probabilities)
    total_sum = sum(probabilities);
    left_sum = 0;
    min_diff = inf;
    split_index = 0;

    for i = 1:length(probabilities)
        left_sum = left_sum + probabilities(i);
        right_sum = total_sum - left_sum;
        diff = abs(left_sum - right_sum);

        if diff < min_diff
            min_diff = diff;
            split_index = i;
        end
```

```
    end
end
```

```
Symbol  Probability  Code
A       0.50         0
B       0.20         10
C       0.20         110
D       0.05         1110
E       0.05         1111
```

Published with MATLAB® R2024b