```matlab
% To Perform Shannon-Fano coding technique

clc;
clear;
close all;

% Define a probability series
symbols = ['A', 'B', 'C', 'D', 'E'];
probabilities = [0.5, 0.2, 0.2, 0.05, 0.05];

% Sort probabilities in descending order
[probabilities, index] = sort(probabilities, 'descend');
symbols = symbols(index); % Rearrange symbols accordingly

% Initialize code dictionary
global codes;
codes = cell(1, length(probabilities));

% Call the Shannon-Fano function
shannon_fano(probabilities, symbols, 1, length(probabilities), '');

% Display the result
fprintf('Symbol\tProbability\tCode\n');
for i = 1:length(symbols)
    fprintf('%c\t%.2f\t\t%s\n', symbols(i), probabilities(i), codes{i});
end

% Shannon-Fano Encoding Function
function shannon_fano(probabilities, symbols, start_idx, end_idx, code)
    global codes;  % Access global codes array

    % Base case: If one symbol left, assign the final code
    if start_idx == end_idx
        codes{start_idx} = code;
        return;
    end

    % Find partition point where probabilities are balanced
    total = sum(probabilities(start_idx:end_idx));
    partial_sum = 0;
    split_index = start_idx;

    for i = start_idx:end_idx
        partial_sum = partial_sum + probabilities(i);
        if partial_sum >= total / 2
            split_index = i;
            break;
        end
    end

    % Assign '0' and '1' to the divided groups
    for i = start_idx:split_index
```

```
        codes{i} = [code '0'];
    end
    for i = split_index+1:end_idx
        codes{i} = [code '1'];
    end

    % Recursively call function on both partitions
    shannon_fano(probabilities, symbols, start_idx, split_index, [code '0']);
    shannon_fano(probabilities, symbols, split_index+1, end_idx, [code '1']);
end
```

```
Symbol    Probability    Code
A    0.50        0
B    0.20        100
C    0.20        101
D    0.05        110
E    0.05        111
```

Published with MATLAB® R2024b