```
%Shannon Fano Coding
%Created by Boda Pavan

clear all;
close all;
clc;

symbols = {'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H'};
frequencies = [2048, 2048, 2048, 2048, 819, 819, 3277, 3277];

% Compute the probabilities of each symbol
probabilities = [];
probabilities = [probabilities, frequencies / sum(frequencies)];

% Sort the probabilities in descending order and rearrange symbols accordingly
[probabilities, idx] = sort(probabilities, 'descend');
symbols = symbols(idx);

% Initialize cell array to store Huffman codes and array for code lengths
codes = cell(size(symbols));
code_lengths = zeros(size(symbols));

% Initialize stack for iterative Shannon-Fano encoding
stack = {{symbols, probabilities, ''}};

% Iteratively perform Shannon-Fano encoding
while ~isempty(stack)
    data = stack{end};
    stack(end) = [];
    s = data{1};
    p = data{2};
    pre = data{3};

    % If only one symbol remains, assign its final code
    if length(s) == 1
        idx = strcmp(symbols, s{1});
        codes{idx} = pre;
        code_lengths(idx) = length(pre);
        continue;
    end

    % Determine the best split point based on cumulative probability sum
    split = find(cumsum(p) >= sum(p) / 2, 1);

    % Push the two split groups onto the stack with appropriate prefixes
    stack{end+1} = {s(1:split), p(1:split), strcat(pre, '0')}; % Left branch
    stack{end+1} = {s(split+1:end), p(split+1:end), strcat(pre, '1')}; %
Right branch
end

% Display the resulting symbol codes and their lengths
fprintf('Symbol\tCode\tLength\n');
```

```
for i = 1:length(symbols)
    fprintf('%s\t%s\t%d\n', symbols{i}, codes{i}, code_lengths(i));
end
```

```
Symbol    Code    Length
G    000    3
H    001    3
A    01    2
B    100    3
C    101    3
D    110    3
E    1110    4
F    1111    4
```

Published with MATLAB® R2024b