```
function HuffmanEncoding()

clc;
clear all;
close all;

% Step 1: Define symbols and probabilities
symbols = {'A','B','C','D','E'};
p = [0.30 0.25 0.20 0.15 0.10];

% Step 2: Create initial nodes
nodes = [];

for i = 1:length(symbols)
    node.symbol = symbols{i};
    node.prob   = p(i);
    node.left   = [];
    node.right  = [];
    nodes = [nodes node];
end

% Step 3: Build Huffman Tree
while length(nodes) > 1

% Sort nodes by probability (ascending)
    [~, idx] = sort([nodes.prob]);
    nodes = nodes(idx);
% Take two smallest nodes
    n1 = nodes(1);
    n2 = nodes(2);
% Create parent node
    parent.symbol = '';
    parent.prob   = n1.prob + n2.prob;
    parent.left   = n1;
    parent.right  = n2;
% Remove first two nodes and add parent
    nodes = nodes(3:end);
    nodes(end+1) = parent;
end

tree = nodes(1);

% Step 4: Generate Huffman Codes
codebook = containers.Map();
codebook = buildCodes(tree, '', codebook);

% Step 5: Display Huffman Codes
disp('Huffman Codes:');
for i = 1:length(symbols)
    s = symbols{i};
    fprintf('Symbol %s  ->  Code %s\n', s, codebook(s));
end
```

```
% Step 6: Average Code Length
avgLen = 0;
for i = 1:length(symbols)
    avgLen = avgLen + p(i) * length(codebook(symbols{i}));
end

fprintf('\nAverage Code Length = %.3f bits/symbol\n', avgLen);

end


function codebook = buildCodes(node, code, codebook)

% If leaf node # assign code
if isempty(node.left) && isempty(node.right)
    codebook(node.symbol) = code;
    return;
end

% Go left # add 0
if ~isempty(node.left)
    codebook = buildCodes(node.left, [code '0'], codebook);
end

% Go right # add 1
if ~isempty(node.right)
    codebook = buildCodes(node.right, [code '1'], codebook);
end

end
```

```
Huffman Codes:
Symbol A  ->  Code 11
Symbol B  ->  Code 01
Symbol C  ->  Code 00
Symbol D  ->  Code 101
Symbol E  ->  Code 100

Average Code Length = 2.250 bits/symbol
```

*Published with MATLAB® R2021a*