

```

% Name :Nenavath Dhanush Rathod
% BT23ECE120

% Huffman Coding
clear all;
close all;
clc;
% Input symbols and probabilities
symbols = {'A','B','C','D','E'};
prob = [0.4 0.2 0.2 0.1 0.1];

% Create node structure
nodes = struct('symbols',{},{}, 'prob',{},{}, 'left',{},{}, 'right',{},{});

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

% Build Huffman Tree
while length(nodes) > 1

    % Sort nodes by probability (ascending)
    [~, idx] = sort([nodes.prob]);
    nodes = nodes(idx);

    % Take two least probable nodes
    leftNode = nodes(1);
    rightNode = nodes(2);

    % Create new merged node
    newNode.symbols = [leftNode.symbols rightNode.symbols];
    newNode.prob = leftNode.prob + rightNode.prob;
    newNode.left = leftNode;
    newNode.right = rightNode;

    % Remove first two nodes
    nodes(1:2) = [];

    % Add merged node back
    nodes(end+1) = newNode;
end

% Generate Huffman Codes
codes = containers.Map;
generateCodes(nodes, '', codes);

% Display Results
disp(' Symbol      Probability      Huffman Code');

for i = 1:length(symbols)
    fprintf('      %s          %.2f          %s\n', ...
        symbols{i}, prob(i), codes(symbols{i}));
end

```

```
end

% Recursive function to assign codes
function generateCodes(node, code, codes)
    if isempty(node.left) && isempty(node.right)
        codes(node.symbols) = code;
    else
        generateCodes(node.left, [code '0'], codes);
        generateCodes(node.right, [code '1'], codes);
    end
end
```

Symbol	Probability	Huffman Code
A	0.40	11
B	0.20	00
C	0.20	01
D	0.10	100
E	0.10	101