```
% Written by Rajveer Singh (BT23ECE108)
% Huffman Coding using Manual Probability Merging
% (No toolbox functions used)

clc;
clear;
close all;
```

Input symbols and probabilities

```
symbols = {'m1','m2','m3','m4','m5','m6','m7','m8'};
P = [0.22 0.18 0.15 0.12 0.10 0.09 0.08 0.06];

% Normalize probabilities
P = P / sum(P);
```

Display symbols and probabilities

```
disp('Given Symbols and Probabilities');
for i = 1:length(symbols)
    fprintf('%s : %.3f\n', symbols{i}, P(i));
end
```

```
Given Symbols and Probabilities
m1 : 0.220
m2 : 0.180
m3 : 0.150
m4 : 0.120
m5 : 0.100
m6 : 0.090
m7 : 0.080
m8 : 0.060
```

Manual Huffman Merging Process

```
disp(' ');
disp('Huffman Coding – Manual Probability Merging Steps');

temp_probs = P;
temp_syms  = symbols;
step = 1;

while length(temp_probs) > 1

    % Sort probabilities in ascending order
    [temp_probs, idx] = sort(temp_probs);
    temp_syms = temp_syms(idx);

    % Display merging step
    fprintf('Step %d: Merge %.3f (%s) + %.3f (%s) = %.3f\n', ...
        step, temp_probs(1), temp_syms{1}, ...
        temp_probs(2), temp_syms{2}, ...
        temp_probs(1) + temp_probs(2));

    % Merge two smallest probabilities
    temp_probs(2) = temp_probs(1) + temp_probs(2);
    temp_syms{2}  = [temp_syms{1} '+' temp_syms{2}];

    % Remove the smallest entry
    temp_probs(1) = [];
    temp_syms(1)  = [];

    step = step + 1;
end
```

```
Huffman Coding – Manual Probability Merging Steps
Step 1: Merge 0.060 (m8) + 0.080 (m7) = 0.140
Step 2: Merge 0.090 (m6) + 0.100 (m5) = 0.190
Step 3: Merge 0.120 (m4) + 0.140 (m8+m7) = 0.260
Step 4: Merge 0.150 (m3) + 0.180 (m2) = 0.330
Step 5: Merge 0.190 (m6+m5) + 0.220 (m1) = 0.410
Step 6: Merge 0.260 (m4+m8+m7) + 0.330 (m3+m2) = 0.590
Step 7: Merge 0.410 (m6+m5+m1) + 0.590 (m4+m8+m7+m3+m2) = 1.000
```

Final Huffman Codes (Manually Assigned)

```
disp(' ');
disp('Final Huffman Codes (Prefix-Free)');

codes = {
    'm1', '00';
    'm2', '01';
    'm3', '100';
    'm4', '101';
    'm5', '1100';
    'm6', '1101';
    'm7', '1110';
    'm8', '1111'
};

for i = 1:size(codes,1)
    fprintf('%s   P=%.3f   Code=%s\n', ...
        codes{i,1}, P(i), codes{i,2});
end
```

```
Final Huffman Codes (Prefix-Free)
m1   P=0.220   Code=00
m2   P=0.180   Code=01
m3   P=0.150   Code=100
m4   P=0.120   Code=101
m5   P=0.100   Code=1100
m6   P=0.090   Code=1101
```

```
m7   P=0.080   Code=1110
m8   P=0.060   Code=1111
```

Average Code Length

```
disp(' ');
disp('Average Code Length Calculation');

code_lengths = cellfun(@length, codes(:,2));
L_avg = sum(P .* code_lengths');

fprintf('Average Code Length = %.4f bits/symbol\n', L_avg);
```

```
Average Code Length Calculation
Average Code Length = 2.9300 bits/symbol
```

Entropy Calculation

```
H = -sum(P .* log2(P));
fprintf('Source Entropy = %.4f bits/symbol\n', H);
```

```
Source Entropy = 2.8834 bits/symbol
```

Verification

```
if L_avg >= H
    disp('Verification: Average code length ≥ Entropy (Valid Huffman Code)');
else
    disp('Verification: Condition not satisfied');
end
```

```
Verification: Average code length ≥ Entropy (Valid Huffman Code)
```

Published with MATLAB® R2025b