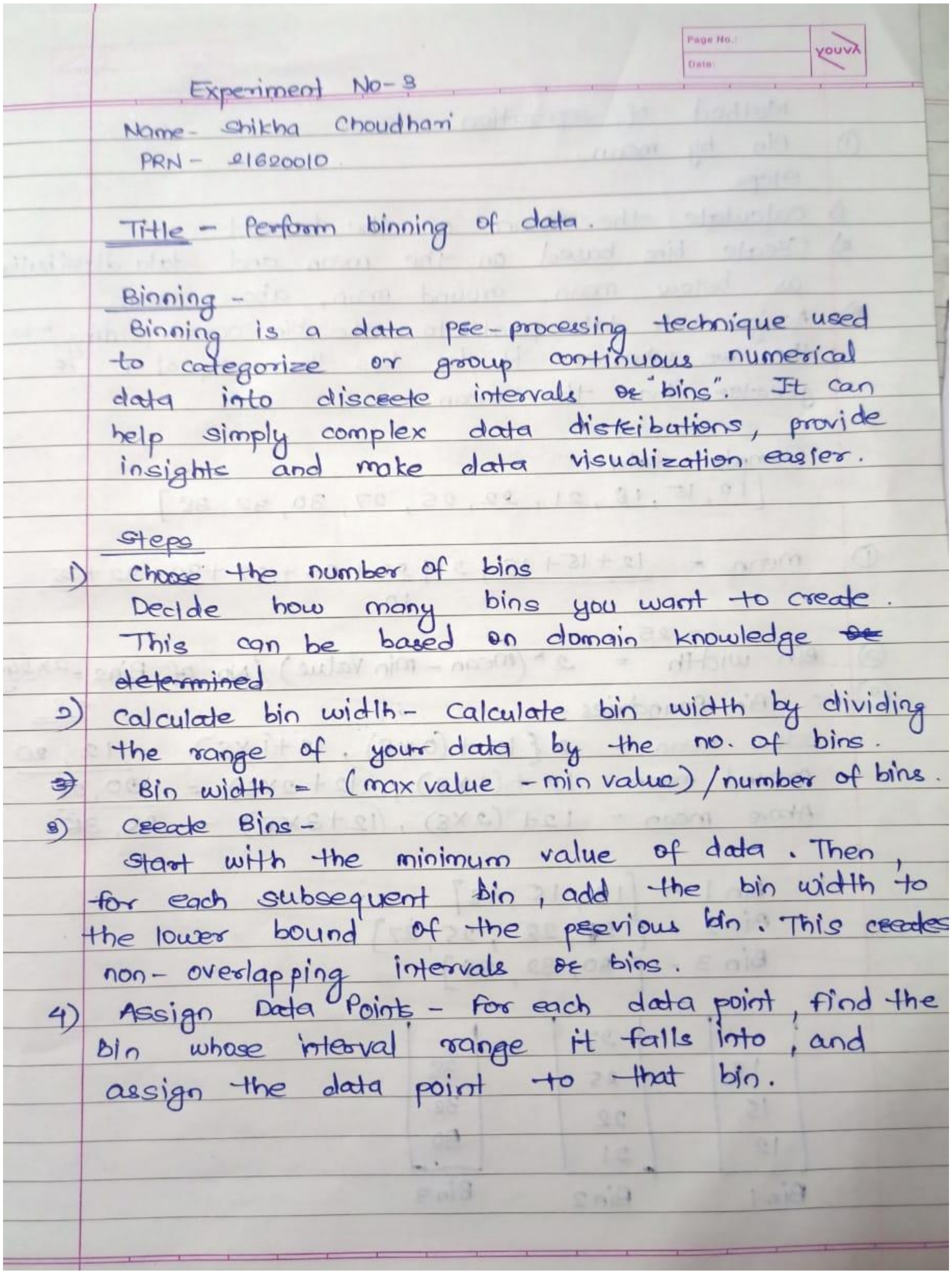
Experiment- 3

Name - Shikha Sanjay Choudhari
PRN- 21620010

Title- Perform Binning of data.



Experiment No- 3

Name- Shikha Choudhari
PRN - 21620010

Title - Perform binning of data.

Binning -

Binning is a data pre-processing technique used to categorize or group continuous numerical data into discrete intervals or "bins". It can help simply complex data distributions, provide insights and make data visualization easier.

Steps

1) Choose the number of bins
Decide how many bins you want to create. This can be based on domain knowledge ~~or~~ ~~determined~~

2) Calculate bin width - Calculate bin width by dividing the range of your data by the no. of bins.

~~3)~~ Bin width = (max value - min value)/number of bins.

3) Create Bins -
Start with the minimum value of data. Then, for each subsequent bin, add the bin width to the lower bound of the previous bin. This creates non-overlapping intervals or bins.

4) Assign Data Points - For each data point, find the bin whose interval range it falls into, and assign the data point to that bin.

Method of seperation.

① Bin by mean.

Steps

1) calculate the mean of the dataset

2) Ceeate bins based on the mean and data disteibution as below mean, around mean, above mean.

3) Assign each data point to the bin corresponding to ~~the we~~ whether it is less than, equal to or geeater than the mean.

Sample data -

[12, 15, 18, 21, 22, 25, 27, 30, 32, 38]

① $\text{mean} = \dfrac{12+15+18+21+22+25+27+30+32+38}{10}$

$= 25$

② Bin width $= 2 * (\text{Mean} - \text{Min Value}) / \text{No. of Bins} = 2 \times \dfrac{(24-12)}{3}$

$= 8$

② ∴ Bin Boundries -

Below mean - $\{12 + (0 * 8),\ 12 + (1 \times 8) = 12, 20$

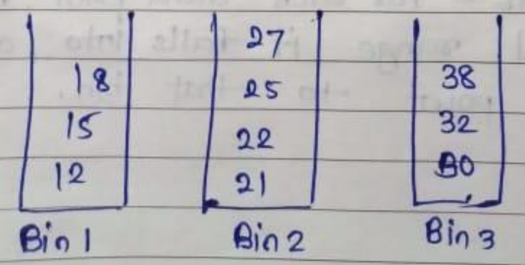
Around mean - $12 + (1 \times 8),\ (12 + 2 \times 8) = 20, 28$

Above mean - $12 + (2 \times 8),\ (12 + 3 \times 8) = 28, 36$

∴ Bin 1 - [12, 15, 18]

Bin 2 - [21, 22, 25, 27]

Bin 3 - [30, 32, 38]



* Binning by Boundary

Steps

i) Determine the minimum and maximum values in the dataset.

ii) Calculate the range of the data {.

iii) Choose the no. of bins you want to create.

iv) Calculate the bin width by dividing the range by no. of bins.

v) Ceeate bins based on the bin width & assign data points to the appropriate bins based on their values.

Sample data - [12, 15, 18, 21, 22, 25, 27, 30, 32, 38]

① min- Max value
Min value = 12
Max value = 38

② Calculate range
Range = Max. value - Min value
= 38-12
= 26

③ No. of bins = 3

④ Bin width $= \frac{\text{Range by}}{\text{No. of bins}} = \frac{26}{3} = 8.67$

⑤ Bin boundaries -
Bin 1 ⇒ Values <= Min value + Bin width = 12+8.67
Values <= 20.67

Bin 2 : Values > Min value + Bin width and
<= Min value + 2* Bin width

$20.67 + 8.67 = 29.34$

$20.67 < \text{Values} \le 29.34$

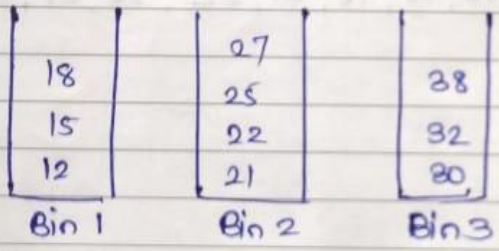
Bin 3 - Values > Min value + 2* Bin width =
Values > 29.34

∴ Data seperation in bins -

Bin 1 - [12, 15, 18]

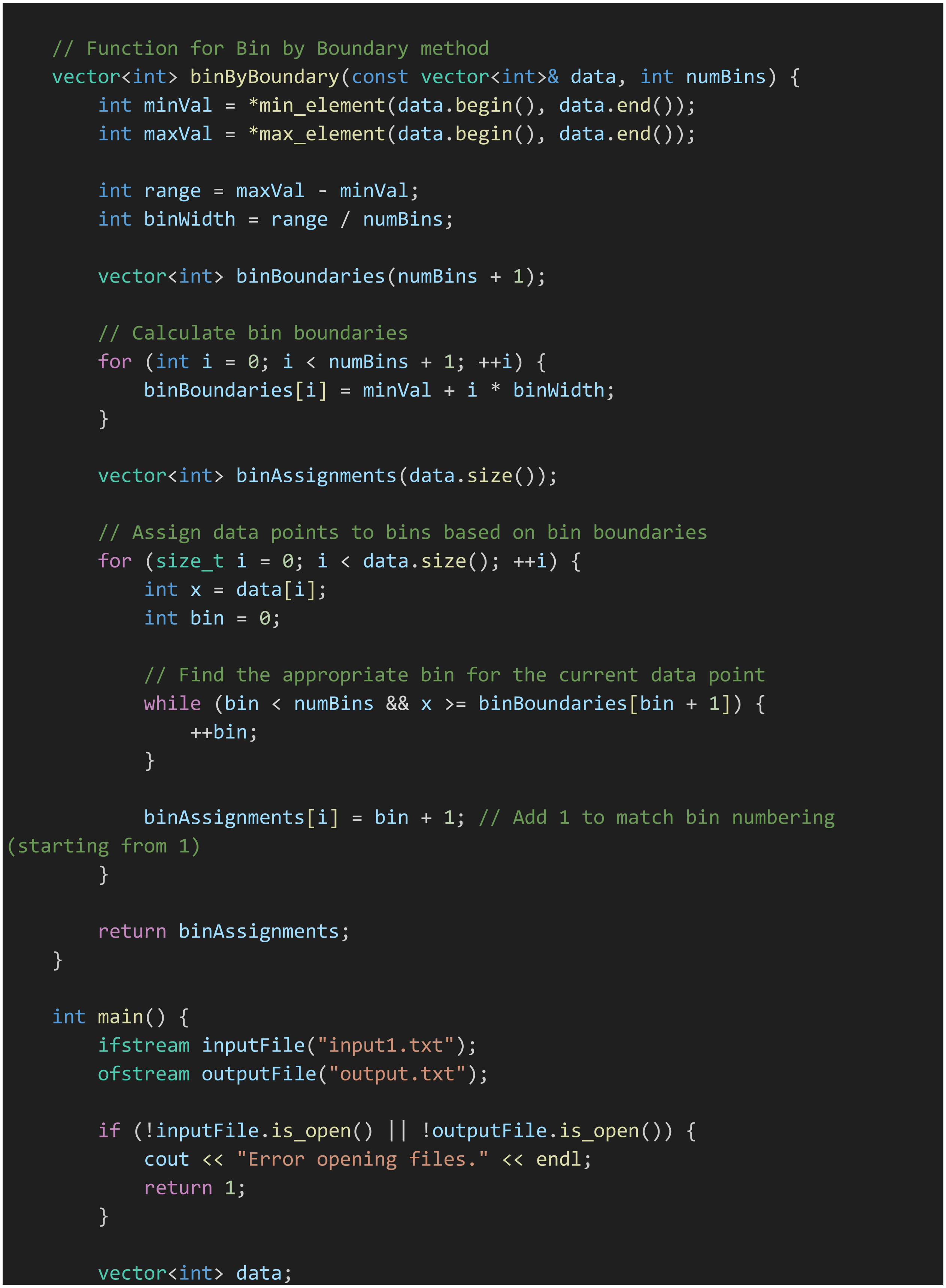
Bin 2 - [21, 22, 25, 27]

Bin 3 - [30, 32, 38]

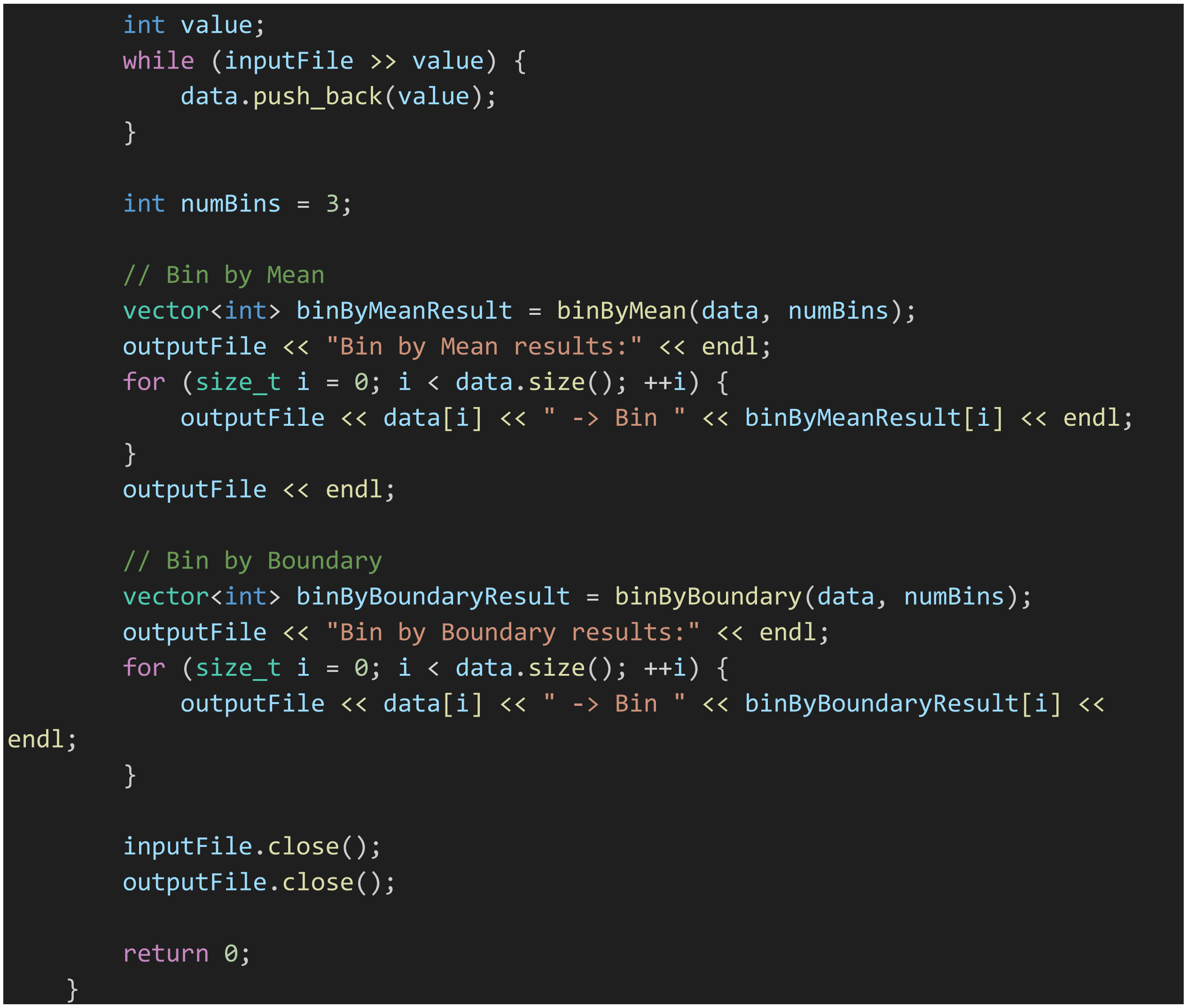


- **Code**

```
#include <iostream>
    #include <fstream>
    #include <vector>
    #include <algorithm>

    using namespace std;

    // Function for Bin by Mean method
    vector<int> binByMean(const vector<int>& data, int numBins) {
        int sum = 0;
        for (int x : data) {
            sum += x;
        }
        double mean = static_cast<double>(sum) / data.size();

        vector<int> binBoundaries(numBins + 1);
        int binWidth = 0;

        // Calculate bin boundaries and bin width
        binWidth = (int)(2 * (mean - data.front()) / numBins);

        for (int i = 0; i < numBins + 1; ++i) {
            binBoundaries[i] = data.front() + i * binWidth;
        }

        vector<int> binAssignments(data.size());

        // Assign data points to bins based on mean-based boundaries
        for (size_t i = 0; i < data.size(); ++i) {
            int x = data[i];
            int bin = numBins - 1; // Initialize to last bin

            // Find the appropriate bin for the current data point
            while (bin >= 0 && x < binBoundaries[bin]) {
                --bin;
            }

            binAssignments[i] = bin + 1; // Add 1 to match bin numbering
(starting from 1)
        }

        return binAssignments;
    }
```

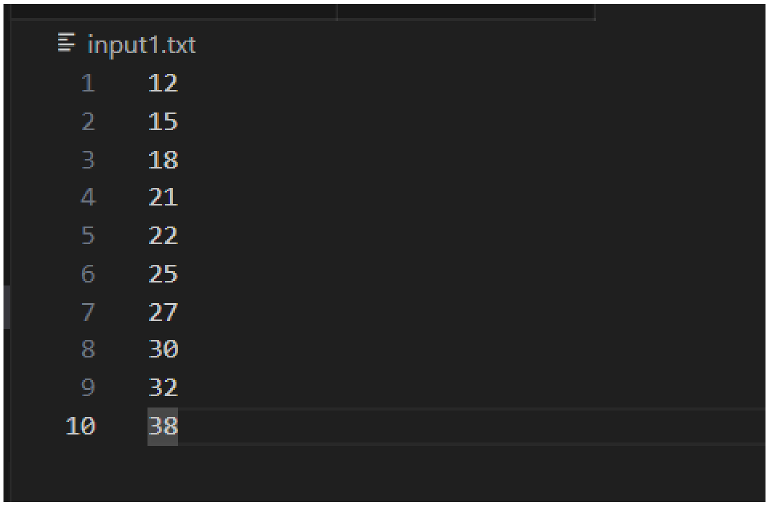


```
    // Function for Bin by Boundary method
    vector<int> binByBoundary(const vector<int>& data, int numBins) {
        int minVal = *min_element(data.begin(), data.end());
        int maxVal = *max_element(data.begin(), data.end());

        int range = maxVal - minVal;
        int binWidth = range / numBins;

        vector<int> binBoundaries(numBins + 1);

        // Calculate bin boundaries
        for (int i = 0; i < numBins + 1; ++i) {
            binBoundaries[i] = minVal + i * binWidth;
        }

        vector<int> binAssignments(data.size());

        // Assign data points to bins based on bin boundaries
        for (size_t i = 0; i < data.size(); ++i) {
            int x = data[i];
            int bin = 0;

            // Find the appropriate bin for the current data point
            while (bin < numBins && x >= binBoundaries[bin + 1]) {
                ++bin;
            }

            binAssignments[i] = bin + 1; // Add 1 to match bin numbering
(starting from 1)
        }

        return binAssignments;
    }

    int main() {
        ifstream inputFile("input1.txt");
        ofstream outputFile("output.txt");

        if (!inputFile.is_open() || !outputFile.is_open()) {
            cout << "Error opening files." << endl;
            return 1;
        }

        vector<int> data;
```



```
        int value;
        while (inputFile >> value) {
            data.push_back(value);
        }

        int numBins = 3;

        // Bin by Mean
        vector<int> binByMeanResult = binByMean(data, numBins);
        outputFile << "Bin by Mean results:" << endl;
        for (size_t i = 0; i < data.size(); ++i) {
            outputFile << data[i] << " -> Bin " << binByMeanResult[i] << endl;
        }
        outputFile << endl;

        // Bin by Boundary
        vector<int> binByBoundaryResult = binByBoundary(data, numBins);
        outputFile << "Bin by Boundary results:" << endl;
        for (size_t i = 0; i < data.size(); ++i) {
            outputFile << data[i] << " -> Bin " << binByBoundaryResult[i] <<
endl;
        }

        inputFile.close();
        outputFile.close();

        return 0;
    }
```

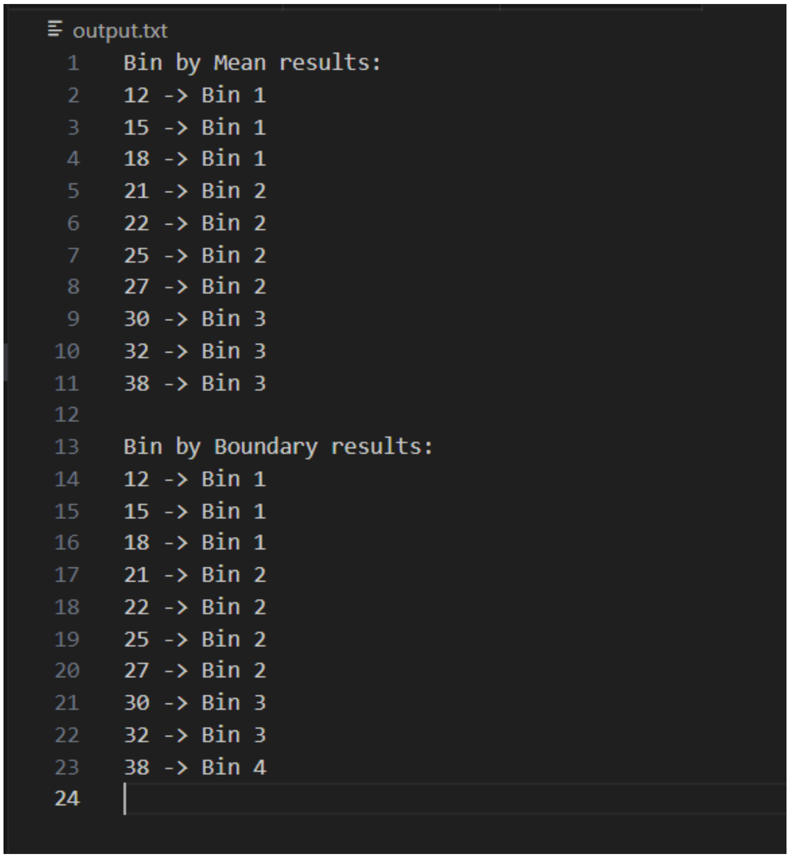
- **Input data**



input1.txt

```
12
15
18
21
22
25
27
30
32
38
```

- **Output**



```
Bin by Mean results:
12 -> Bin 1
15 -> Bin 1
18 -> Bin 1
21 -> Bin 2
22 -> Bin 2
25 -> Bin 2
27 -> Bin 2
30 -> Bin 3
32 -> Bin 3
38 -> Bin 3

Bin by Boundary results:
12 -> Bin 1
15 -> Bin 1
18 -> Bin 1
21 -> Bin 2
22 -> Bin 2
25 -> Bin 2
27 -> Bin 2
30 -> Bin 3
32 -> Bin 3
38 -> Bin 4
```