Create a JSON File:

- Create a file (e.g., `input_data.json`) with your JSON input.

```
{
   "keys": {
      "n": 4,
      "k": 3
   },
   "1": {
      "base": "10",
      "value": "4"
   },
   "2": {
      "base": "2",
      "value": "111"
   },
   "3": {
      "base": "10",
      "value": "12"
   },
   "6": {
      "base": "4",
      "value": "213"
   }
```

}

Java Code: (Sample test – 1)

```
import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;


public class ShamirSecretSharing {


    // Function to decode the value based on its base

    public static int decodeBaseValue(int base, String value) {

        return Integer.parseInt(value, base);

    }


    // Function to calculate the constant term (c) using Lagrange interpolation

    public static double lagrangeInterpolation(List<Integer> xValues, List<Integer> yValues, int
k) {

        double constantTerm = 0.0;

        for (int j = 0; j < k; j++) {

            double term = yValues.get(j);  // Start with y_j value

            for (int m = 0; m < k; m++) {

                if (m != j) {

                    term *= (0.0 - xValues.get(m)) / (xValues.get(j) - xValues.get(m));  // Lagrange
interpolation formula
```

```
            }
        }
        constantTerm += term;
    }
    return constantTerm;  // Return the calculated constant term
}

// Function to find the constant term based on given data
public static void findConstantTerm(Map<String, Object> data) {
    @SuppressWarnings("unchecked")
    Map<String, Integer> keys = (Map<String, Integer>) data.get("keys");  // Safely cast to Map<String, Integer>
    int n = keys.get("n");
    int k = keys.get("k");

    List<Integer> xValues = new ArrayList<>();
    List<Integer> yValues = new ArrayList<>();

    // Iterate through all the points
    for (String key : data.keySet()) {
        if (key.equals("keys")) continue;  // Skip the 'keys' entry

        @SuppressWarnings("unchecked")
```

```
        Map<String, String> point = (Map<String, String>) data.get(key);  // Safely cast to Map<String, String>
        int base = Integer.parseInt(point.get("base"));
        String value = point.get("value");

        int x = Integer.parseInt(key);  // x is the key of the point
        int y = decodeBaseValue(base, value);  // y is the decoded value

        xValues.add(x);
        yValues.add(y);

        if (xValues.size() == k) break;  // Stop after collecting k points
      }

      // Calculate the constant term using Lagrange interpolation
      double constantTerm = lagrangeInterpolation(xValues, yValues, k);
      System.out.println((int) Math.round(constantTerm));  // Print the rounded constant term
    }

    // Main function to define the input and call the function
    public static void main(String[] args) {
      // Manually created input map for the given problem
      Map<String, Object> inputData = new HashMap<>();
```

```
// Create the "keys" map
Map<String, Integer> keys = new HashMap<>();
keys.put("n", 4);
keys.put("k", 3);
inputData.put("keys", keys);

// Create points for the polynomial roots
Map<String, String> point1 = new HashMap<>();
point1.put("base", "10");
point1.put("value", "4");
inputData.put("1", point1);

Map<String, String> point2 = new HashMap<>();
point2.put("base", "2");
point2.put("value", "111");
inputData.put("2", point2);

Map<String, String> point3 = new HashMap<>();
point3.put("base", "10");
point3.put("value", "12");
inputData.put("3", point3);

Map<String, String> point6 = new HashMap<>();
```

```
        point6.put("base", "4");
        point6.put("value", "213");
        inputData.put("6", point6);

        // Call the function to find the constant term
        findConstantTerm(inputData);
    }
}
```

How to Run:

1. **Set up the `org.json` library**: If you're using Maven, add the following dependency to your `pom.xml`:

   ```
   xml
   Copy code
   <dependency>
       <groupId>org.json</groupId>
       <artifactId>json</artifactId>
       <version>20210307</version>
   </dependency>
   ```

   Or download the JAR file and include it in your project's classpath.

2. **Compile and Run**:
   - Compile the Java code:

     ```
     javac -cp .:json.jar ShamirSecretSharing.java
     ```

   - Run the compiled Java code:

     ```
     java -cp .:json.jar ShamirSecretSharing
     ```

3. Replace `json.jar` with the actual path to the JSON library JAR file if needed.

Expected Output:

Running the above code with the provided JSON input should output:

```
3
```

```
{
    "keys": {
        "n": 9,
        "k": 6
    },
    "1": {
        "base": "10",
        "value": "28735619723837"
    },
    "2": {
        "base": "16",
        "value": "1A228867F0CA"
    },
    "3": {
        "base": "12",
        "value": "32811A4AA0B7B"
    },
    "4": {
        "base": "11",
        "value": "917978721331A"
    },
    "5": {
        "base": "16",
        "value": "1A22886782E1"
    },
    "6": {
        "base": "10",
        "value": "28735619654702"
    },
    "7": {
        "base": "14",
        "value": "71AB5070CC4B"
    },
    "8": {
        "base": "9",
        "value": "122662581541670"
    },
    "9": {
        "base": "8",
        "value": "642121030037605"
    }
}
```

Java Code: (Sample test – 2)

```
import java.math.BigInteger;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class ShamirSecretSharing {

    public static BigInteger decodeBaseValue(int base, String value) {
        return new BigInteger(value, base);
    }

    public static BigInteger lagrangeInterpolation(List<BigInteger> xValues,
List<BigInteger> yValues, int k) {
        BigInteger c = BigInteger.ZERO;
        for (int j = 0; j < k; j++) {
            BigInteger term = yValues.get(j);
            for (int m = 0; m < k; m++) {
                if (m != j) {
                    BigInteger x_j = xValues.get(j);
                    BigInteger x_m = xValues.get(m);
                    BigInteger numerator = BigInteger.ZERO.subtract(x_m);
                    BigInteger denominator = x_j.subtract(x_m);
                    term = term.multiply(numerator).divide(denominator);
                }
            }
            c = c.add(term);
        }
        return c;
    }

    public static void findConstantTerm(Map<String, Object> data) {
        Map<String, Integer> keys = (Map<String, Integer>) data.get("keys");
        int n = keys.get("n");
        int k = keys.get("k");

        List<BigInteger> xValues = new ArrayList<>();
        List<BigInteger> yValues = new ArrayList<>();

        for (String key : data.keySet()) {
            if (key.equals("keys")) continue;

            Map<String, String> point = (Map<String, String>) data.get(key);
            int base = Integer.parseInt(point.get("base"));
            String value = point.get("value");

            BigInteger x = new BigInteger(key);
            BigInteger y = decodeBaseValue(base, value);

            xValues.add(x);
            yValues.add(y);

            if (xValues.size() == k) break;
        }

        BigInteger constantTerm = lagrangeInterpolation(xValues, yValues, k);
        System.out.println(constantTerm);
```

```
}

public static void main(String[] args) {
    Map<String, Object> inputData = new HashMap<>();

    Map<String, Integer> keys = new HashMap<>();
    keys.put("n", 9);
    keys.put("k", 6);
    inputData.put("keys", keys);

    Map<String, String> point1 = new HashMap<>();
    point1.put("base", "10");
    point1.put("value", "28735619723837");
    inputData.put("1", point1);

    Map<String, String> point2 = new HashMap<>();
    point2.put("base", "16");
    point2.put("value", "1A228867F0CA");
    inputData.put("2", point2);

    Map<String, String> point3 = new HashMap<>();
    point3.put("base", "12");
    point3.put("value", "32811A4AA0B7B");
    inputData.put("3", point3);

    Map<String, String> point4 = new HashMap<>();
    point4.put("base", "11");
    point4.put("value", "917978721331A");
    inputData.put("4", point4);

    Map<String, String> point5 = new HashMap<>();
    point5.put("base", "16");
    point5.put("value", "1A22886782E1");
    inputData.put("5", point5);

    Map<String, String> point6 = new HashMap<>();
    point6.put("base", "10");
    point6.put("value", "28735619654702");
    inputData.put("6", point6);

    Map<String, String> point7 = new HashMap<>();
    point7.put("base", "14");
    point7.put("value", "71AB5070CC4B");
    inputData.put("7", point7);

    Map<String, String> point8 = new HashMap<>();
    point8.put("base", "9");
    point8.put("value", "122662581541670");
    inputData.put("8", point8);

    Map<String, String> point9 = new HashMap<>();
    point9.put("base", "8");
    point9.put("value", "642121030037605");
    inputData.put("9", point9);

    findConstantTerm(inputData);
}
```

```
}
```

Expected Output:

Running the above code with the provided JSON input should output:

```
28735619723846
```