// a) String Compression

// Bonus1 : input =a2b2c1a3c3;

// Output=ab2c1ac3;

To improve memory usage, one approach is to use compression algorithms to reduce the size of the data being stored. One such algorithm is run-length encoding (RLE), which can be used to compress repetitive data by replacing runs of identical values with a count of the number of repetitions and the value itself.

Here's an implementation of RLE compression in Java that can handle the bonus requirements:

```
public class RLE {
    public static String compress(String input) {
        StringBuilder output = new StringBuilder();
        int count = 1;
        char prev = input.charAt(0);

        for (int i = 1; i < input.length(); i++) {
            char current = input.charAt(i);
            if (current == prev) {
                count++;
            } else {
                output.append(prev);
                output.append(count);
                prev = current;
                count = 1;
            }
        }

        // Append the last run
```

```
    output.append(prev);
    output.append(count);

    // Compress further
    return compress2(output.toString());
}

public static String decompress(String input) {
    // Decompress first
    String decompressed = decompress2(input);

    StringBuilder output = new StringBuilder();
    int count = 0;

    for (int i = 0; i < decompressed.length(); i++) {
        char current = decompressed.charAt(i);
        if (Character.isDigit(current)) {
            count = count * 10 + Character.getNumericValue(current);
        } else {
            for (int j = 0; j < count; j++) {
                output.append(current);
            }
            count = 0;
        }
    }

    return output.toString();
}
```

```
public static String compress2(String input) {
    StringBuilder output = new StringBuilder();
    int count = 1;
    char prev = input.charAt(0);

    for (int i = 1; i < input.length(); i++) {
        char current = input.charAt(i);
        if (current == prev) {
            count++;
        } else {
            if (count == 2) {
                output.append(prev);
            } else if (count > 2) {
                output.append(count);
                output.append(prev);
            }
            prev = current;
            count = 1;
        }
    }

    // Append the last run
    if (count == 2) {
        output.append(prev);
    } else if (count > 2) {
        output.append(count);
        output.append(prev);
    }
```

```
    return output.toString();
}

public static String decompress2(String input) {
    StringBuilder output = new StringBuilder();
    int count = 0;

    for (int i = 0; i < input.length(); i++) {
        char current = input.charAt(i);
        if (Character.isDigit(current)) {
            count = count * 10 + Character.getNumericValue(current);
        } else {
            if (count == 0) {
                output.append(current);
            } else if (count == 1) {
                output.append(current);
                count = 0;
            } else if (count > 1) {
                for (int j = 0; j < count - 1; j++) {
                    output.append(current);
                }
                count = 0;
            }
        }
    }

    return output.toString();
}
```

```
public static void main(String[] args) {
    String input = "aabbcaaaccc";
    String compressed = compress(input);
    String decompressed = decompress(compressed);

    System.out.println("Input: " + input);
    System.out.println("Compressed: " + compressed);
    System.out.println("Decompressed: " + decompressed);
```