Table of Contents

File Outputs

a1

Compressing output:

```
234824113a11b5s1f1d1f1s13b148
```

Compression Ratio: 3.28 (Rounded)

Decompressing output:

```
44444444444444444444444222222221111aaaaaaaaaaaaabbbbbb
bbbbbsssssfdfsbbbbbbbbbbbbb88888888888888
```

a2

Compressing output: 94411B3A1B3A

Compression Ratio: 1.75 (Rounded)

Decompressing output: 4444444441111BAAABAAA

Steps taken for verification

The most important step to make sure the program is guaranteed to produce valid and desired output was abstraction of the problem. With that in mind, abstraction is "generalization" of the problem. To make sure the program deals with numbers and big number of repetitions properly, I have developed a "key" which is being stored in memory for each file which has either more than 10 repetitions for numbers or characters. The key format is this

(Location in the original string){character repeated}[how many times repeated];

The key, as told previously told, is being created only for characters that repeated more than 10 times (inclusive). During decompression, when length of the decompressed string become equal to one of the locations in the key string, the program then reads the key and outputs the character in curly braces the same number of times as specified in square brackets. Source code also provides you with explanation. In addition to that, several files (including created by myself) were tested, all produced correct output.