Comp 261 Assignment 5 Report

Question 1:

When reading the War and Peace file, if I search for the word/sentence at the top of the text. Brute Force is usually more faster than KMP however when I find a word all the way to the bottom of the text, by having to read through all the texts KMP tends to get faster at searching and comparing than brute force.

KMP is faster than brute force because KMP and its partial match table thus making it more efficient. However this varies because it highly depends on the text itself and the regularity of patterns that occur in it

Question 2:

```
Character:   code: 110

Character: ! code: 1110000111

Character: " code: 11111010

Character: ' code: 111000010

Character: ( code: 1111101111111

Character: ) code: 011000111000

Character: * code: 11111011010010

Character: , code: 1111111

Character: - code: 100101001

Character: . code: 1110001

Character: / code: 0110001110010101111110

Character: 0 code: 111110110100001

Character: 1 code: 11111011010001

Character: 2 code: 111110110100000

Character: 3 code: 0110001110010111

Character: 4 code: 01100011100101010

Character: 5 code: 0110001110010100

Character: 6 code: 0110001110010110

Character: 7 code: 01100011100111110

Character: 8 code: 01100011100100

Character: 9 code: 01100011100111101

Character: : code: 111000001001
```

```
Character: ; code: 111110110101

Character: = code: 0110001110010101111111

Character: ? code: 1001010100

Character: A code: 011000110

Character: B code: 1110000001

Character: C code: 01100010000

Character: D code: 11111011000

Character: E code: 01100010001

Character: F code: 11100000101

Character: G code: 111110111101

Character: H code: 1110000011

Character: I code: 100101011

Character: J code: 11111011010011

Character: K code: 111110111100

Character: L code: 1111101111110

Character: M code: 1001010101

Character: N code: 1110000000

Character: O code: 01100011101

Character: P code: 011000101

Character: Q code: 0110001110011111 1

Character: R code: 11111011011

Character: S code: 0110001111

Character: T code: 100101000

Character: U code: 01100011100110

Character: V code: 111000001000

Character: W code: 0110001001

Character: X code: 0110001110011110 0

Character: Y code: 111110111110

Character: Z code: 011000111001110

Character: à code: 0110001110010101110

Character: a code: 1000
```

```
Character: b code: 1111100

Character: c code: 101111

Character: ä code: 0110001110010101111010

Character: d code: 10110

Character: e code: 000

Character: f code: 100110

Character: g code: 100100

Character: h code: 0011

Character: i code: 0100

Character: é code: 0110001110010101111011

Character: ê code: 011000111001010110

Character: j code: 11111011001

Character: k code: 0110000

Character: l code: 01101

Character: m code: 101110

Character: n code: 0101

Character: o code: 0111

Character: p code: 1111110

Character: q code: 11111011101

Character: r code: 11110

Character: s code: 0010

Character: t code: 1010

Character: u code: 111011

Character: v code: 1001011

Character: w code: 100111

Character: x code: 1110000110

Character: y code: 011001

Character: z code: 11111011100

Character:  code: 011000111001010111100
```

Final Size:

```
input length:  3258246 bytes
```

output length: 1848598 bytes

Question 3

Ratio = input/output

<u>War and Peace:</u>

input length: 3258246 bytes

output length: 1848598 bytes

Ratio: 1.76

<u>Taisho:</u>

input length: 3649944 bytes

output length: 1542656 bytes

Ratio: 2.37

<u>PI:</u>

input length: 1010003 bytes

output length: 443632 bytes

Ratio: 2.28

Out of the three, Tashio achieves the best compression in size.