Question 1: Write a short summary of the performance you observed using the two search algorithms.

```
Using both version search “sensation” in lenna.txt
Case1: search  ----    616d
Case2:search  ----      6f2e 6772 3977 7d79 0000 1800 4574 74
-------------------------------------------
Brute force search:
Case1:Time spent: 0.005465763    seconds
Case2:Time:0.006099676     seconds
-------------------------------------------
KMP search:
Case1: Time spent: 0.00746547    seconds
Case2:  Time:0.010305107        seconds
```

. Question 2: Report the binary tree of codes your algorithm generates, and the final size of War and Peace after Huffman coding.

BinaryTree.txt

war_and_peace.txt
input length: 3258246 bytes
output length: 1848598 bytes

Question 3: Consider the Huffman coding of war_and_peace.txt, taisho.txt, and pi.txt. Which of these achieves the best encoding, i.e. the best reduction in size? What makes some of the encodings better than others?

war_and_peace.txt
input length: 3258246 bytes
output length: 1848598 bytes
taisho.txt
input length: 3649944 bytes
output length: 1542656 bytes
pi.txt
input length: 1010003 bytes
output length: 443632 bytes

So pi.txt achieve best encoding.

Because, N different messages, log2N bits per message
Pi.txt has the least kinds of messages in the three txt files.
So it the smallest bits per message after encoding

. Question 4: The Lempel-Ziv algorithm has a parameter: the size of the sliding window. On a text of your choice, how does changing the window size affect the quality of the compression?

Compress pi.txt by using Lempel-Ziv

100- size of sliding window
. input length: 1010003 characters
. output length: 2618745 characters

1000 –size of sliding window
. input length: 1010003 characters
. output length: 2232349 characters
. original and decoded texts match!

6000 –size of sliding window
. input length: 1010003 characters
. output length: 2022019 characters
. original and decoded texts match!

Size of windows bigger, then quality of the compression better.

Question 5: What happens if you Huffman encode War and Peace before applying Lempel- Ziv compression to it? Do you get a smaller file size (in characters) overall?

After using Lempel-ziv compress, then using Huffman encode.
input length: 3258227 characters
output length: 6271053 characters
output length: 3153040 bytes

if you Huffman encode War and Peace before applying Lempel- Ziv compression
input length: 3258246 bytes
output length: 1848598 bytes

So direct using Huffman encoding without Lempel-ziv compression works better for war and peace. Txt
Yes, I get a smaller size file