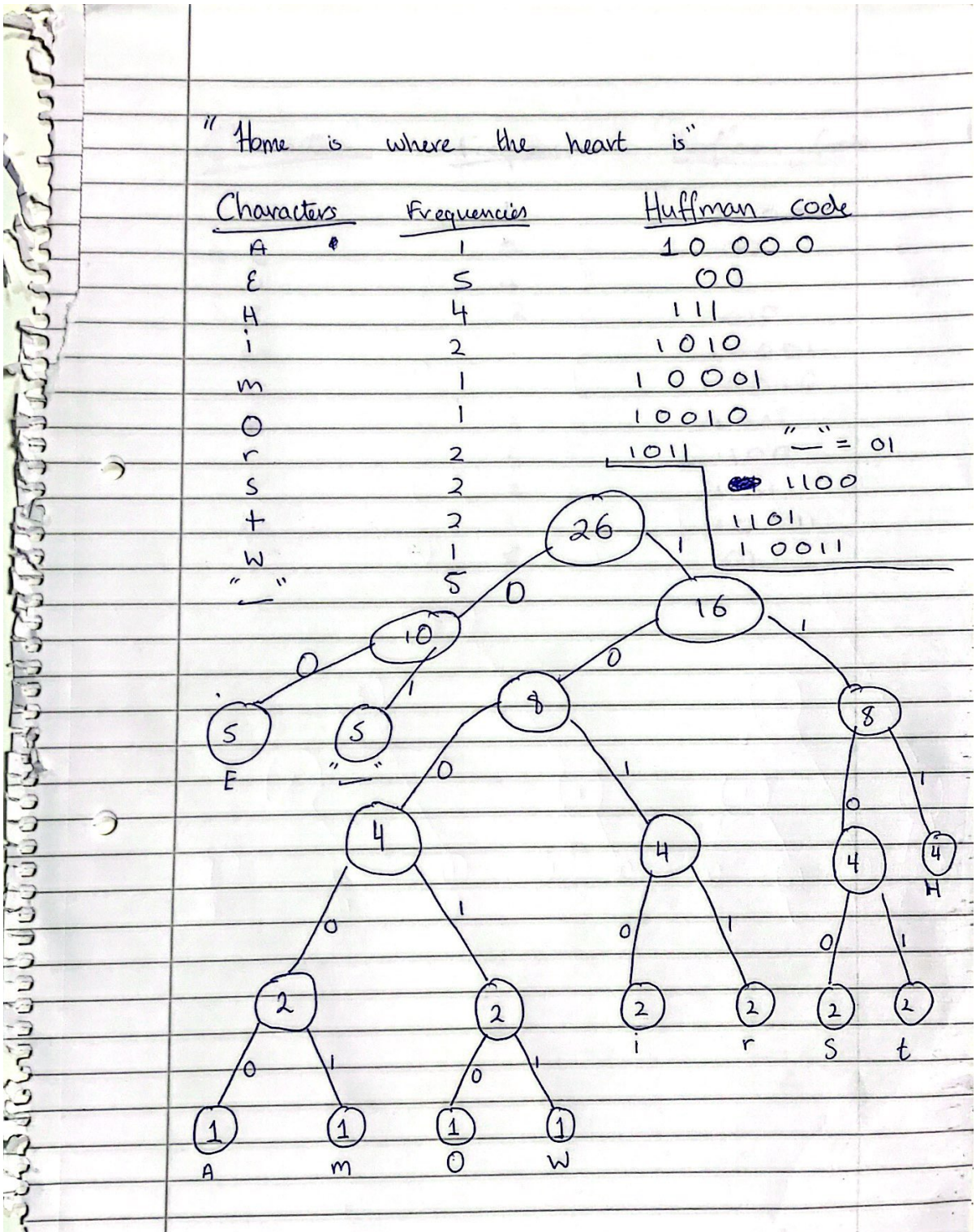
Algorithms
Huffman Compression
Assignment II
20358446
Nathan Betts

Input files to test your implementation:

1. mobydick.txt
2. medTale.txt
3. genomeVirus.txt

4. q32x48.bin



Characters	Frequencies	Huffman Code
A	1	10000
E	5	00
H	4	111
I	2	1010
m	1	10001
O	1	10010
R	2	1011
S	2	1100
T	2	1101
W	1	10011
" "	5	01

File:	Uncompressed Bits	Compressed Bits	Compression Ratio
mobydick.txt	5145680	2921328	5145680 bits
medTale.txt	45056	23912	1.88424
genomeVirus.txt	50008	12576	3.97646
q32x48.bin	1536	816	1.88235

Q2.

File:	Uncompressed Bits	Compressed Bits	After Uncompression
mobydick.txt	5145680	2921328	5145680 bits
medTale.txt	45056	23912	45056 bits
genomeVirus.txt	50008	12576	50008 bits
q32x48.bin	1536	816	1536 bits

After uncompressing the files the after uncompression bits and uncompressed bits are the same, no data was lost.

Q3.

If I apply Huffman compression to an already compressed file, I observed the file grew
Ex.
Q32x48bin compressed bits = 1536 bits
Applying Huffman again
Q32x48bin compressed bits = 2312 bits
After doing some research the reasons for this rise can be attributed to the 1 of the 3 possible (albeit generic) outcomes to compression of a compressed file !)It might get smaller, 2)it might stay the same, and depending on the algorithm,3) I think you might see the file size increase just a bit.

Q4

Q34x48.bin	Huffman	RunLength
Compressed bits	816	1114

After some research, it appears that difference can be attributed to RunLength compressing better by Exploiting long runs of repeated characters.