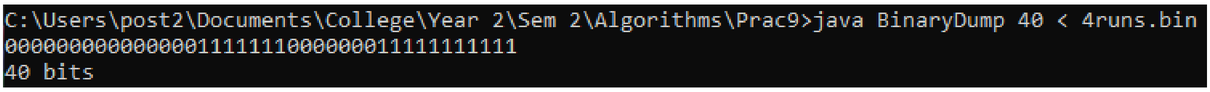
Practical 9 – Binary RunLengthEnconding

Adam Russell – 18328861

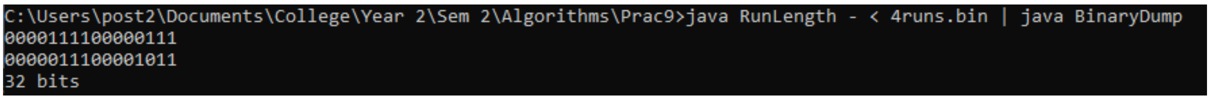
Binary Compression:

1. *Number of bits in 4runs.bin*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 40 < 4runs.bin
0000000000000001111111000000011111111111
40 bits
```

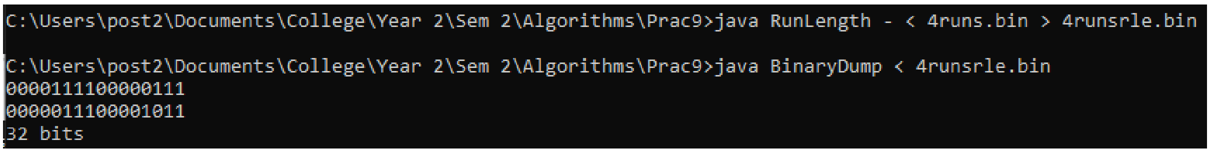
2. *Compressed 4runs.bin*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java RunLength - < 4runs.bin | java BinaryDump
0000111100000111
0000011100001011
32 bits
```

Compression Ratio = 32/40 = 0.8

3. *Output to new file, same bits*

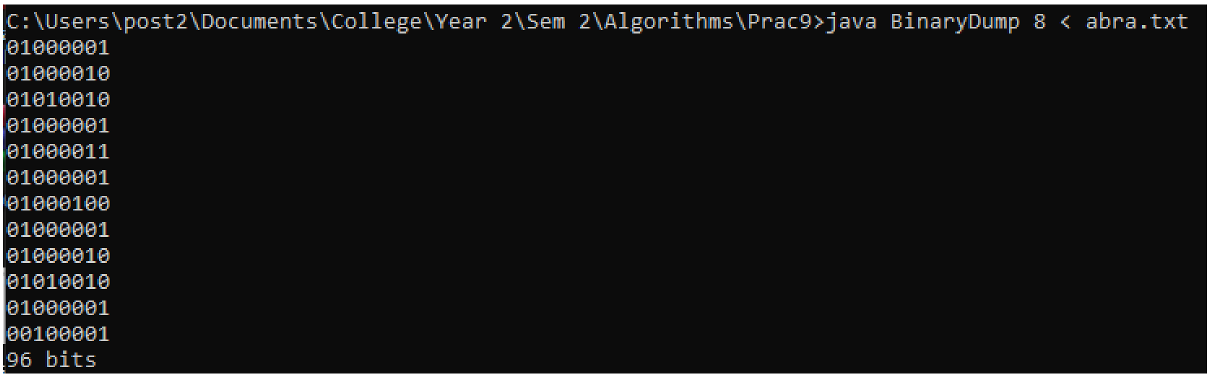


```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java RunLength - < 4runs.bin > 4runsrle.bin

C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump < 4runsrle.bin
0000111100000111
0000011100001011
32 bits
```

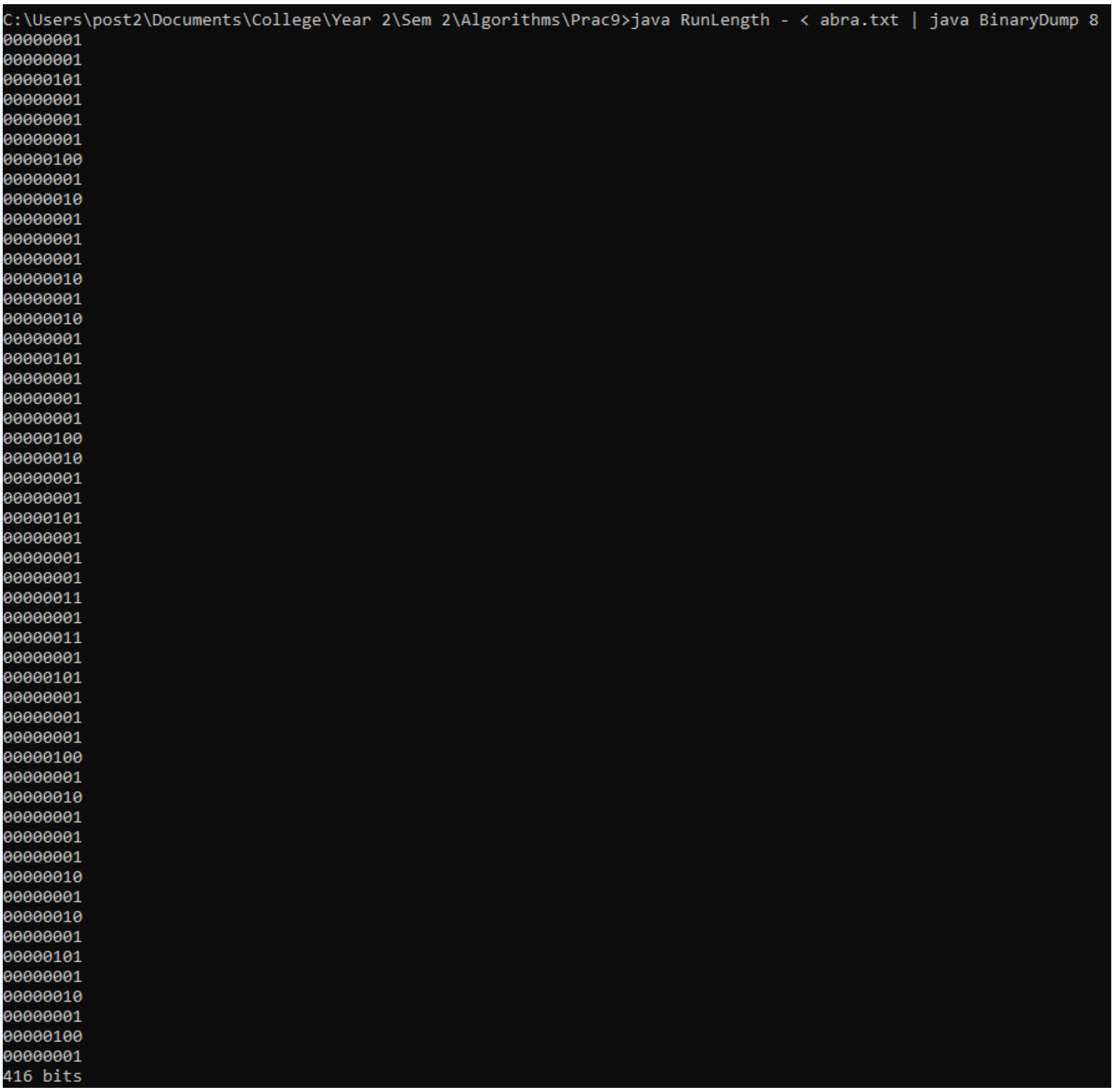
Ascii Compression:

1. *Number of bits in abra.txt*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 8 < abra.txt
01000001
01000010
01010010
01000001
01000011
01000001
01000100
01000001
01000010
01010010
01000001
00100001
96 bits
```

2. *Compress abra.txt*



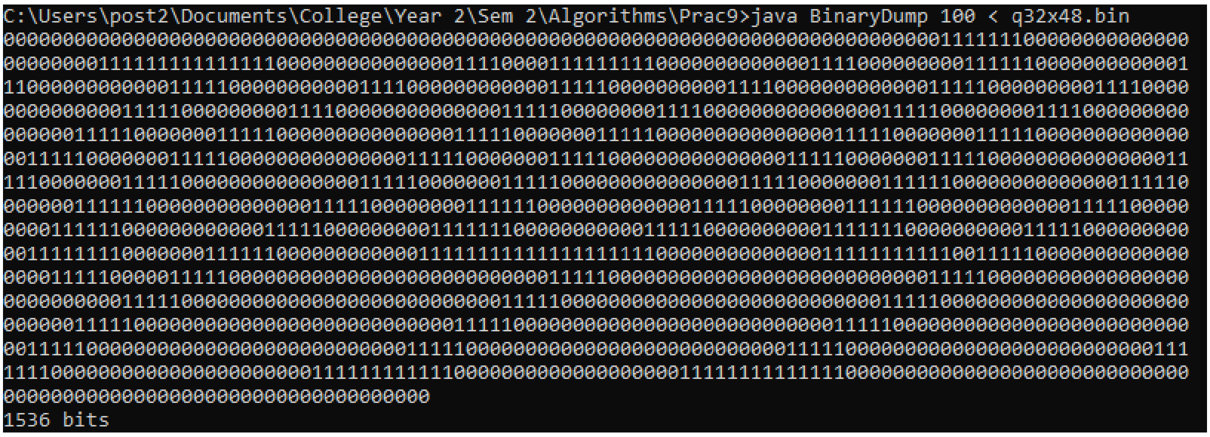
```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java RunLength - < abra.txt | java BinaryDump 8
00000001
00000001
00000101
00000001
00000001
00000001
00000100
00000001
00000010
00000001
00000001
00000001
00000010
00000001
00000010
00000001
00000101
00000001
00000001
00000001
00000100
00000010
00000001
00000001
00000101
00000001
00000001
00000001
00000011
00000001
00000011
00000001
00000101
00000001
00000001
00000001
00000100
00000001
00000010
00000001
00000001
00000001
00000010
00000001
00000010
00000001
00000101
00000001
00000010
00000001
00000100
00000001
416 bits
```

Compression Ratio = 416/96 = 4.33333333

This is probably because there is no runs of the same data, every consecutive character in ABRACADABRA is different, and is therefore hard if not impossible to compress.

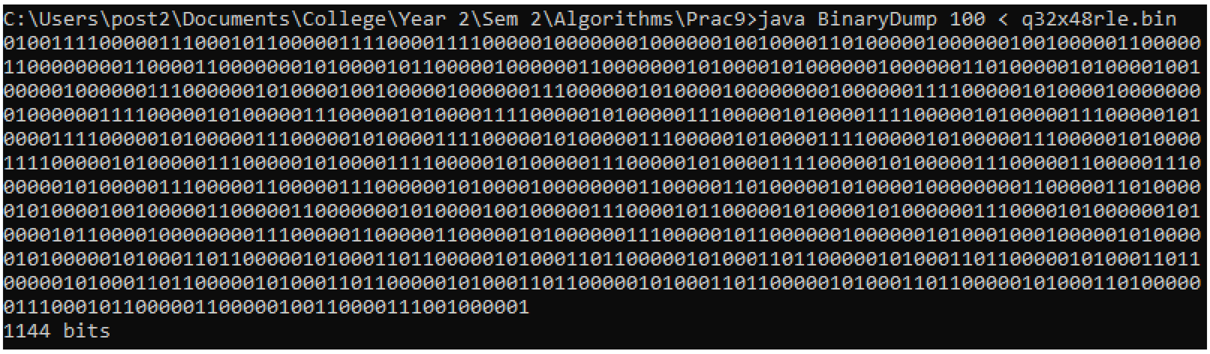
Bitmap Compression:

1. *Number of bits in q32x48.bin*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 100 < q32x48.bin
0000000000000000000000000000000000000000000000000000000000000000000000000000000111111100000000000000
0000000011111111111111100000000000000001111000011111111100000000000000111100000000011111100000000000001
1100000000000001111100000000000001111000000000000011111000000000000111100000000000000111110000000000011110000
0000000001111100000000001111000000000000001111100000000011110000000000000001111100000000011110000000000
0000001111100000000111110000000000000000111110000000111110000000000000000111110000000111110000000000000
0011111000000011111000000000000000011111000000001111100000000000000001111100000000111110000000000000000011
1110000000111110000000000000000111110000000011111000000000000000111110000000111111000000000000000111110
0000000111111000000000000000011111100000000111111000000000000000111110000000011111100000000000000111110000
0000111111000000000000000111111000000000011111100000000000000111111000000000011111100000000000001111100000
0011111111100000001111110000000000000111111111111111111110000000000000001111111111001111110000000000000
0000111110000011111000000000000000000000000000111110000000000000000000000000000111110000000000000000000
0000000001111100000000000000000000000000001111100000000000000000000000000000111110000000000000000000000
0000001111100000000000000000000000000001111100000000000000000000000000001111100000000000000000000000000
0011111000000000000000000000000000001111100000000000000000000000000000111110000000000000000000000000111
1111000000000000000000000000111111111111100000000000000000000111111111111110000000000000000000000000000
000000000000000000000000000000000000
1536 bits
```

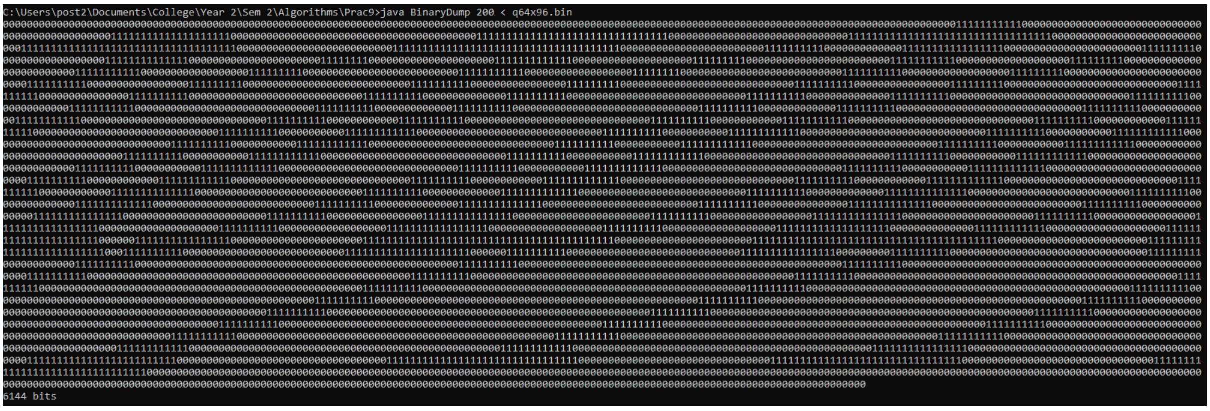
2. *Compress q32x48.bin*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 100 < q32x48rle.bin
0100111100000111000101100000111100001111000001000000010000000100100001101000000100000001001000000110000
1100000000110000110000000101000010110000010000000110000000101000010100000001000000110100000010100001001
00000100000001110000000101000010010000001000000011100000001010000100000000010000000111100000101000010000000
0100000001111000000101000000111000000101000011110000001010000001110000001010000001111000000101000000111000000101
0000111100000010100000011100000010100001111000000101000001111000000101000011110000001010000001110000001010000
1111000000101000000111000000101000011110000001010000011100000010100001111000000101000000111000000110000001110
0000001010000001110000001100000011100000001010000100000000110000011010000001010000100000000110000011010000
0101000010010000001100000110000000101000010010000011100001011000000101000010100000011100001010000000101
00001011000010000000001110000001100000110000010100000001110000010110000001000000010100010001000001010000
0101000000101000110110000001010001101100000010100011011000000101000110110000001010001101100000010100011011
0000010100011011000000101000110110000001010001101100000010100011011000000101000110110000001010001101000000
01110001011000000110000001001100000111001000001
1144 bits
```

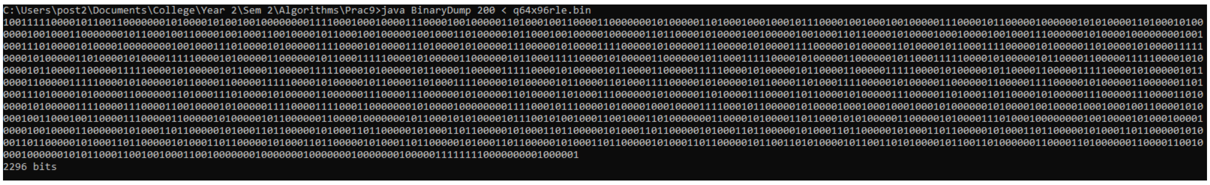
3. Compression Ratio = 1144/1536 = 0.7447

4. *Number of bits in q64x96.bin*



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 200 < q64x96.bin
[illegible]
6144 bits
```

Compress q64x96.bin



```
C:\Users\post2\Documents\College\Year 2\Sem 2\Algorithms\Prac9>java BinaryDump 200 < q64x96rle.bin
10011111000010110011000000001010000101001001000000000111100010001000011100001001000001101000100110000110000000010100000011010001000100010111000010010001001000000111000010110000001000000010101000011010000101
00001001000110000000010110001001100001001000110010000101100010010000001001000110100000010110001001000001000000100000011011000010100001001000000100100011011000010100001000100001001000111000000010100001000000001001
00011101000001010000100000000010010001110100000101000001111000001010000111010000101000000111000000101000011110000001010000001110000001010000111100000010100000011010000101100011111000000101000001101000010100001111
00001010000001101000010100001111100000101000000110000000101100011111000010100000011000000010110001111100001010000001011000011000000111110000101000000101100011111000010100000010110000110000001111100001010000001011
00001011000011000000111110000010100000010110000110000001111100001010000001011000011000000111110000010100000010110000110000001111100000101000000101100001100000011111000010100000010110000110000001111100001010000001011
00001100000011111000001010000001011000011000000111110000101000000101100001101000111100000010100000001011000011010001111000000101000000101100001101000111100000010100000011000000011000000111100000010100000011000000011
00011101000001010000001100000001101000111010000010100000011000000111000001110000001010000001101000011010001110000000101000000110100000111000011011000010100000011100000011010001101100001010000001110000001110000011010
00001010000001111000001110000011001000001010000001111000011110001100000000101000010000000001111000101110000010100001000100001111000101100000010100001000100010001000101000000010100001001000010001000100110000010100
00010011000100110000011100000011000000101000000101100000001100001000000001011000101010000101110010100100011001000110100000000110000101000011011000101010000001100000101000011101000100000000010010000101001000010001
00001001000001100000001010001101100000010100011011000000101000110110000001010001101100000010100011011000000101000110110000001010001101100000010100011011000001010001101100000101000110110000010100011011000001010
00110110000001010001101100000010100011011000000101000110110000001010001101100000010100011011000000101100110101000001011001101010000010110011010100001011001101000000011000011010000000110000110100000001100001100100
000100000001010110001100100100011001000000001000000001000000001000000111111100000000000001000001
2296 bits
```

Compression Ratio = 2296/6144 = 0.3737

5. The compression ratio is most likely higher for the larger file as there is probably longer runs of consecutives 1’s or 0’s allowing for more overall compression.