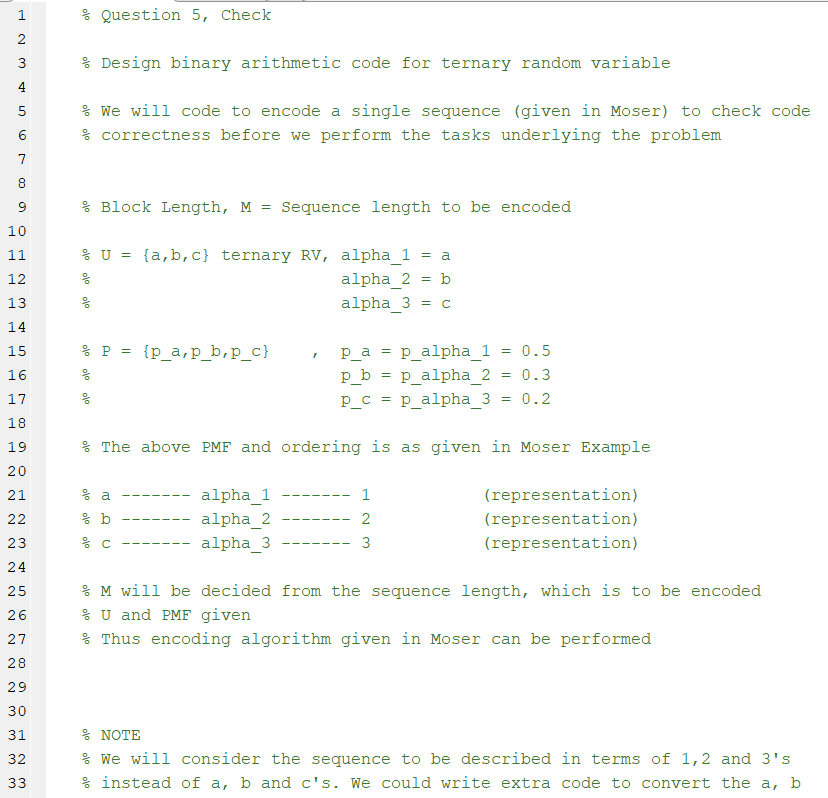
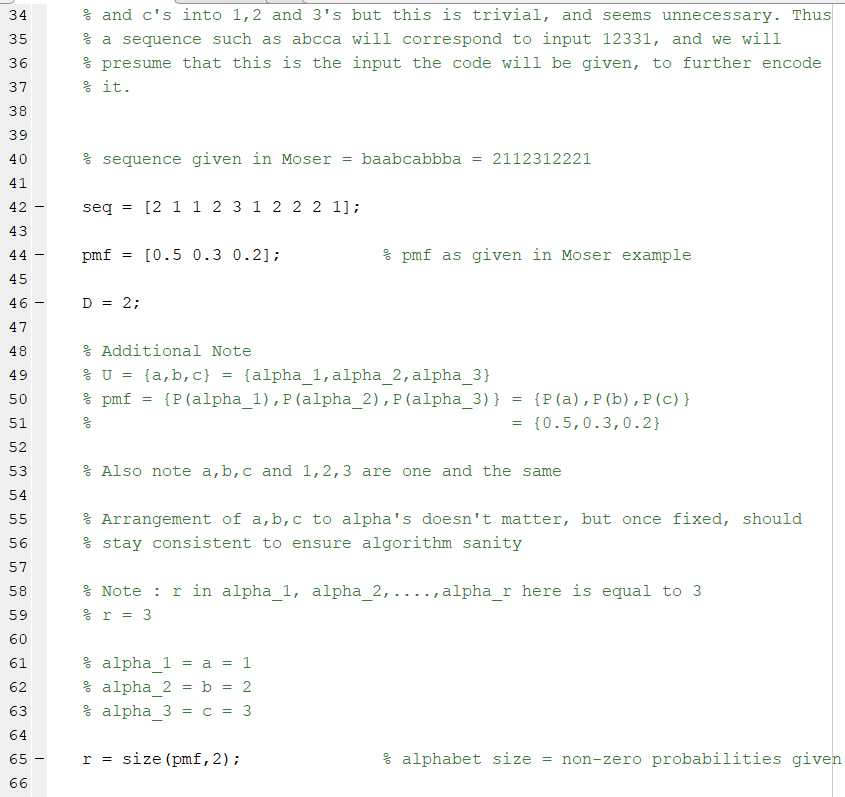
HW 8

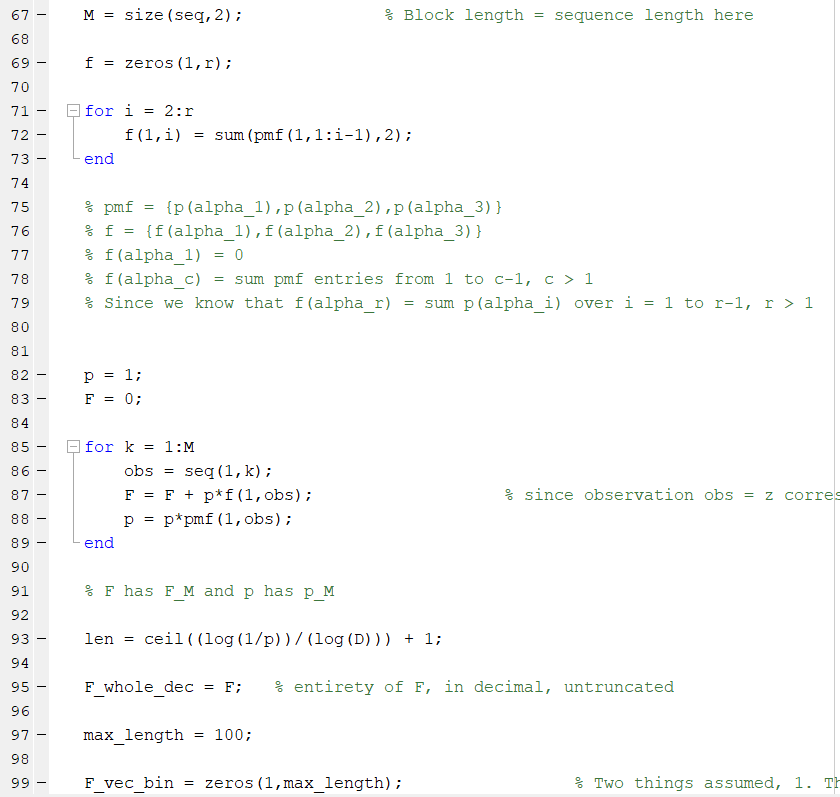
**Question 5**

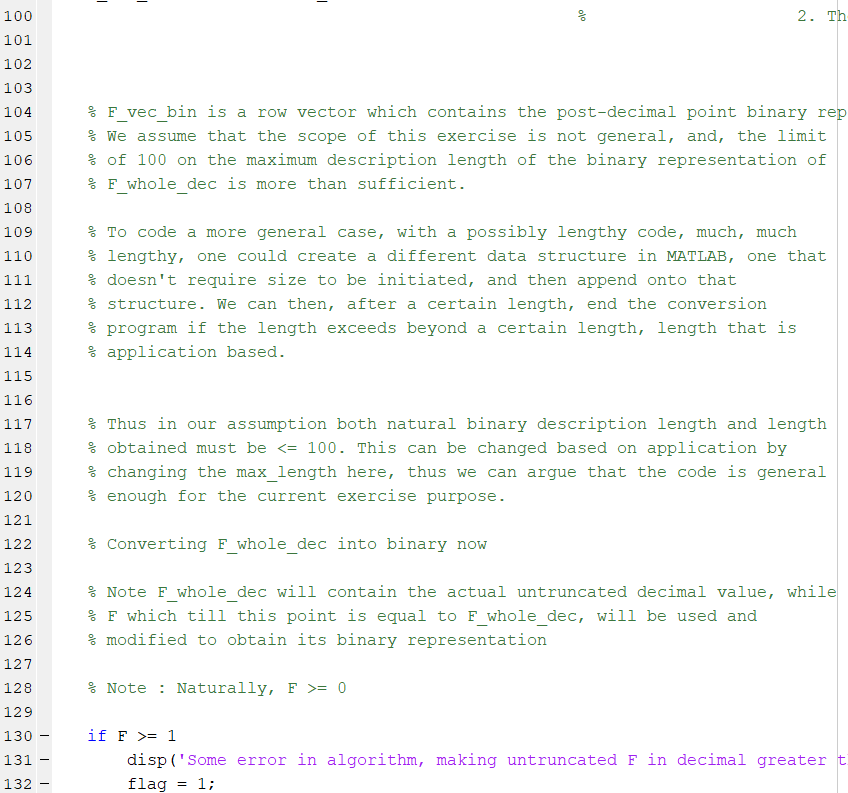
**Checking Portion with Moser Example**

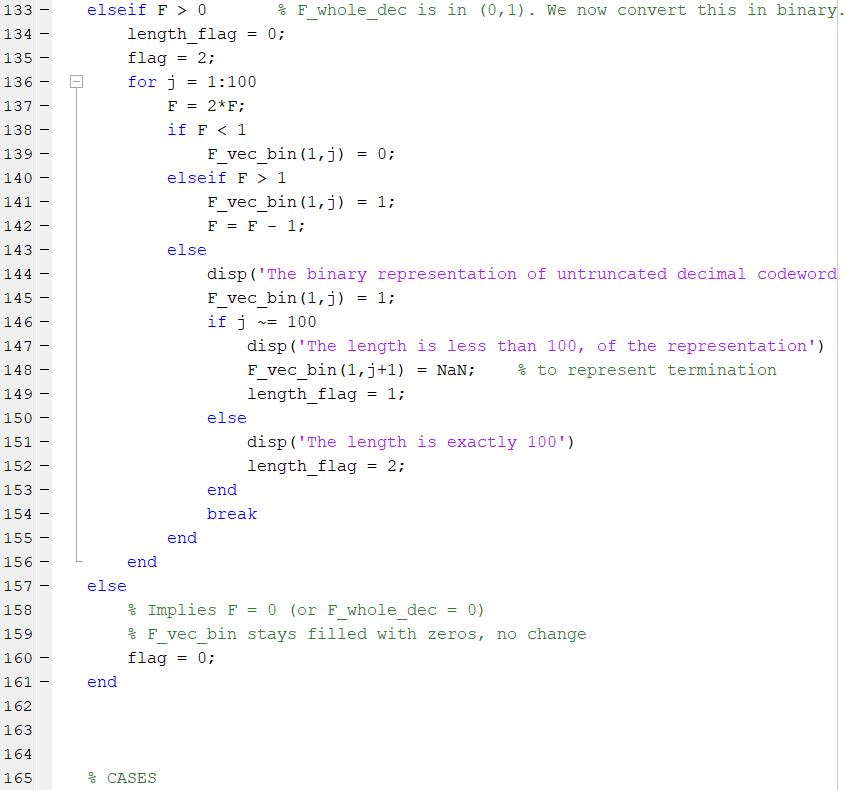
Code

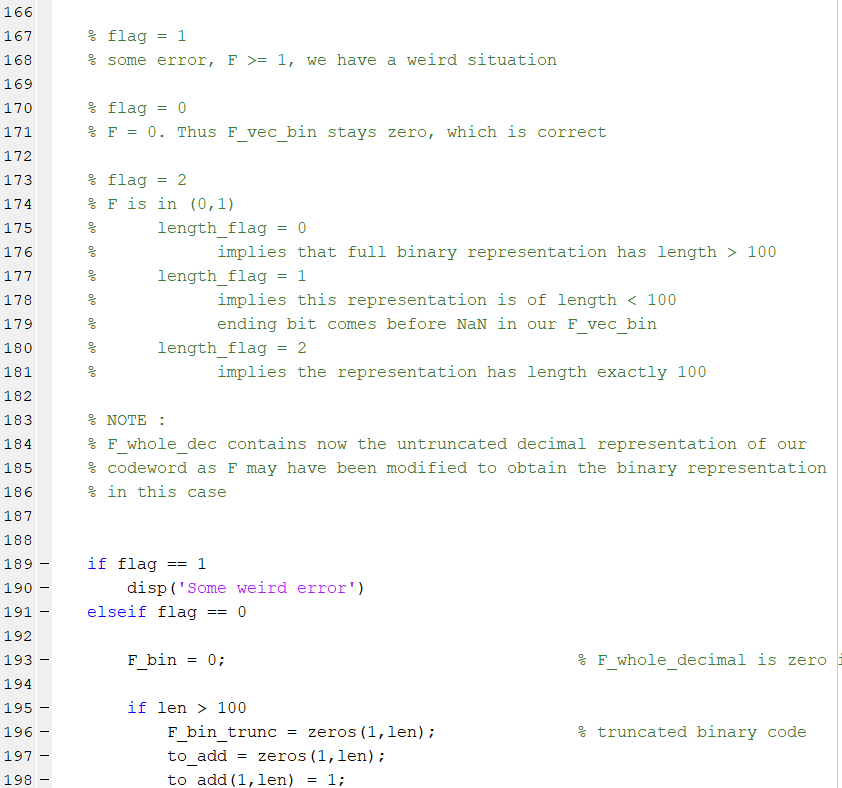


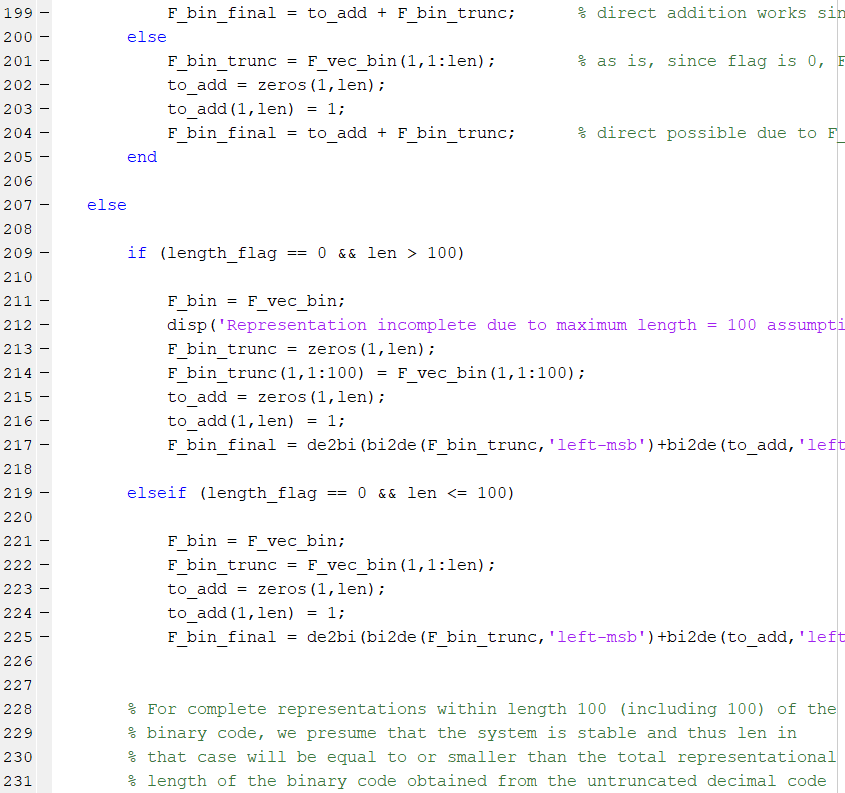


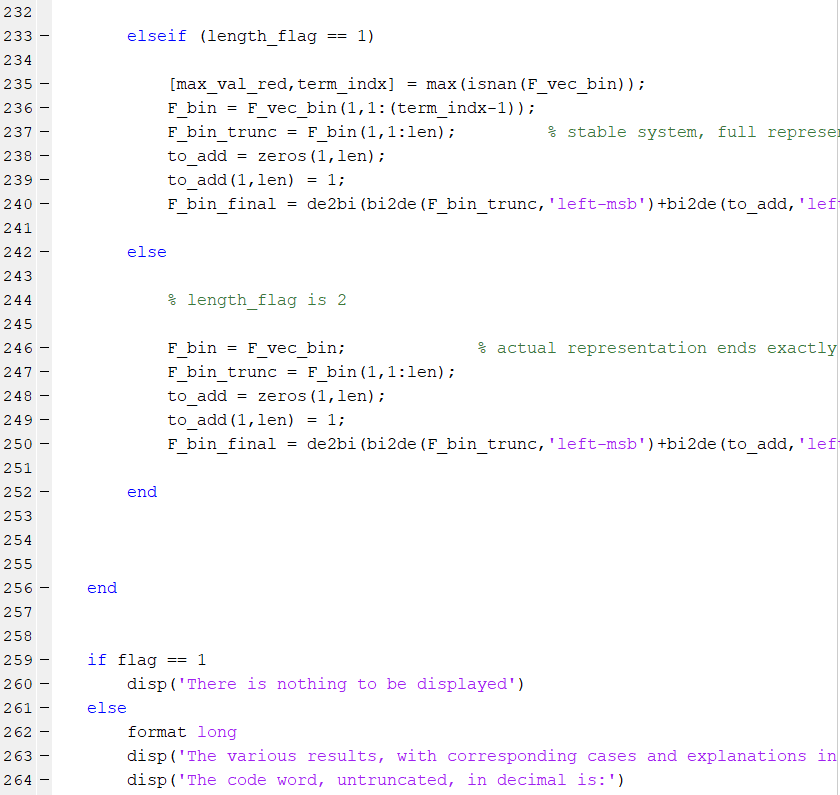


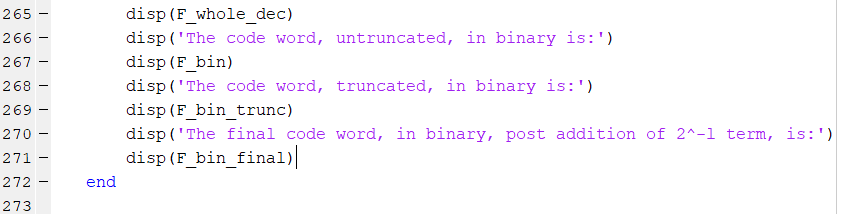










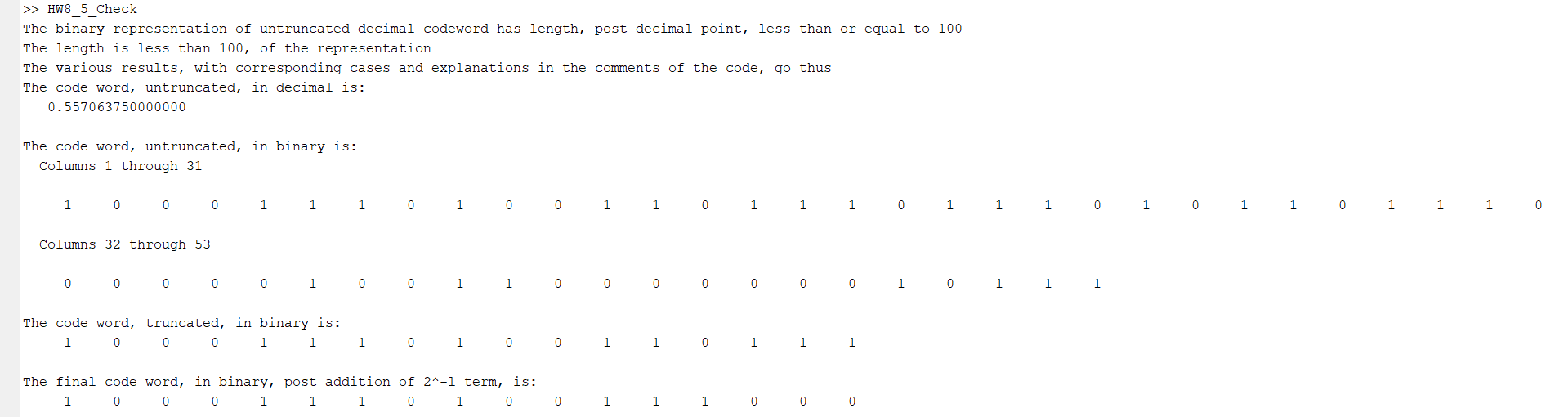


The code has been explained in detail in the various comments.

This code takes care of a huge range of possibly unsavory situations, due to which the code length increased a little bit, despite the simplicity of the arithmetic coding algorithm.

When run, the results agreed with those given in Moser, markedly.

Results of the Check Code



Above snap is unclear, so we copy paste the result tab.

>> HW8\_5\_Check

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The various results, with corresponding cases and explanations in the comments of the code, go thus

The code word, untruncated, in decimal is:

0.557063750000000

The code word, untruncated, in binary is:

Columns 1 through 31

1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 0

Columns 32 through 53

0 0 0 0 0 1 0 0 1 1 0 0 0 0 0 0 0 1 0 1 1 1

The code word, truncated, in binary is:

1 0 0 0 1 1 1 0 1 0 0 1 1 0 1 1 1

The final code word, in binary, post addition of 2^-l term, is:

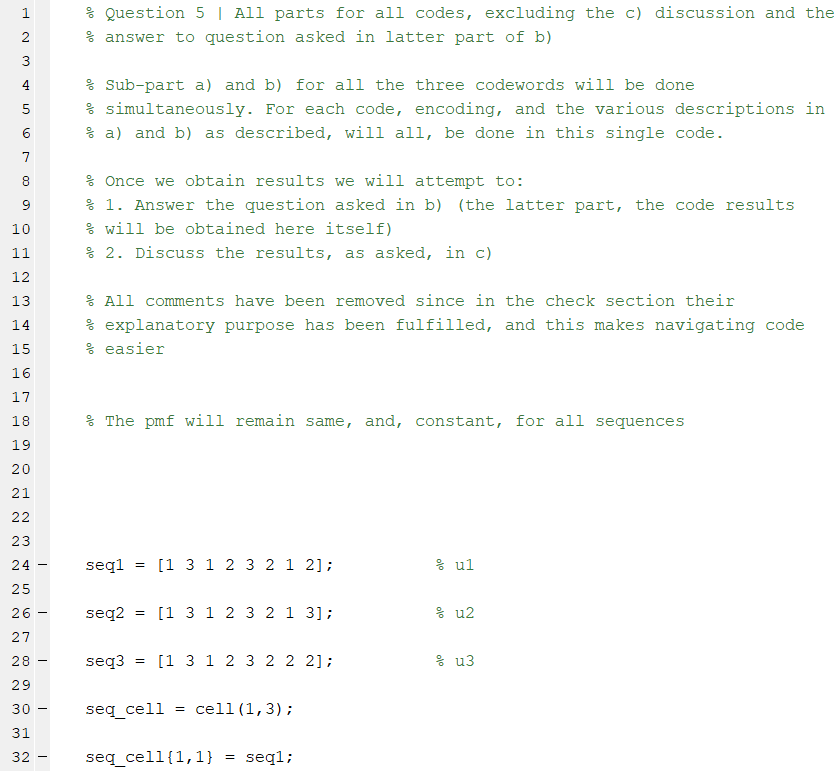
1 0 0 0 1 1 1 0 1 0 0 1 1 1 0 0 0

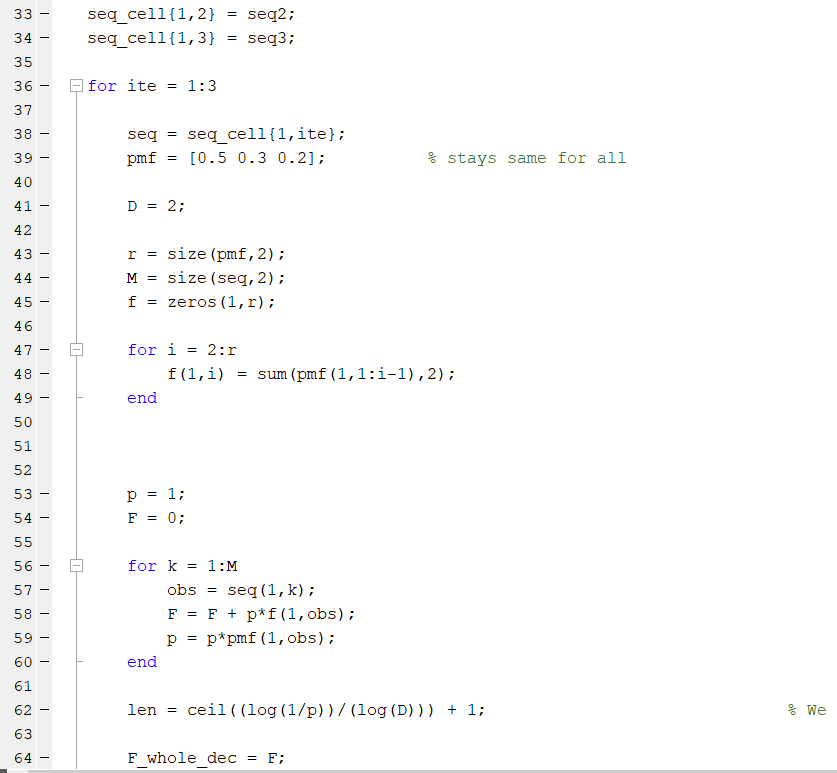
As can be seen from Moser text, the final code word obtained via the code and via Moser is the same, proving that the code works correctly.

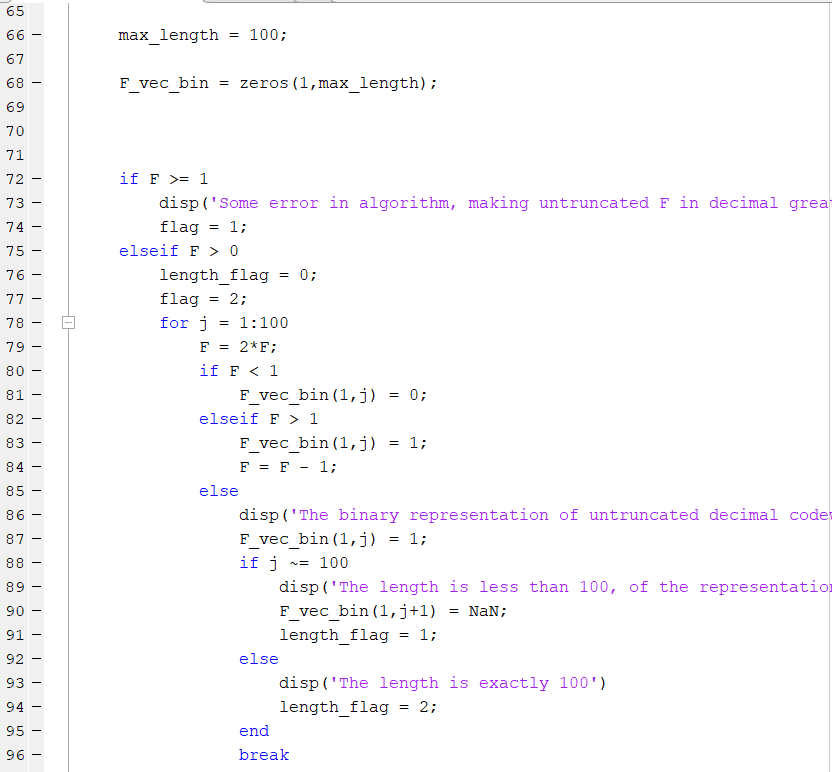
**SUB-PARTS aand bfor all sequence u1, u2 and u3**

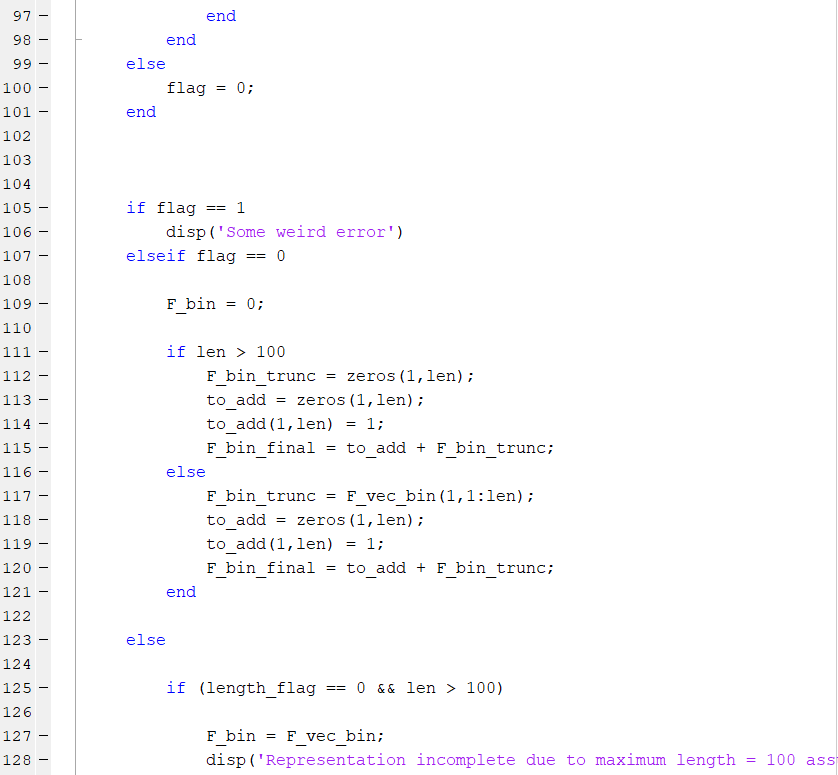
Code

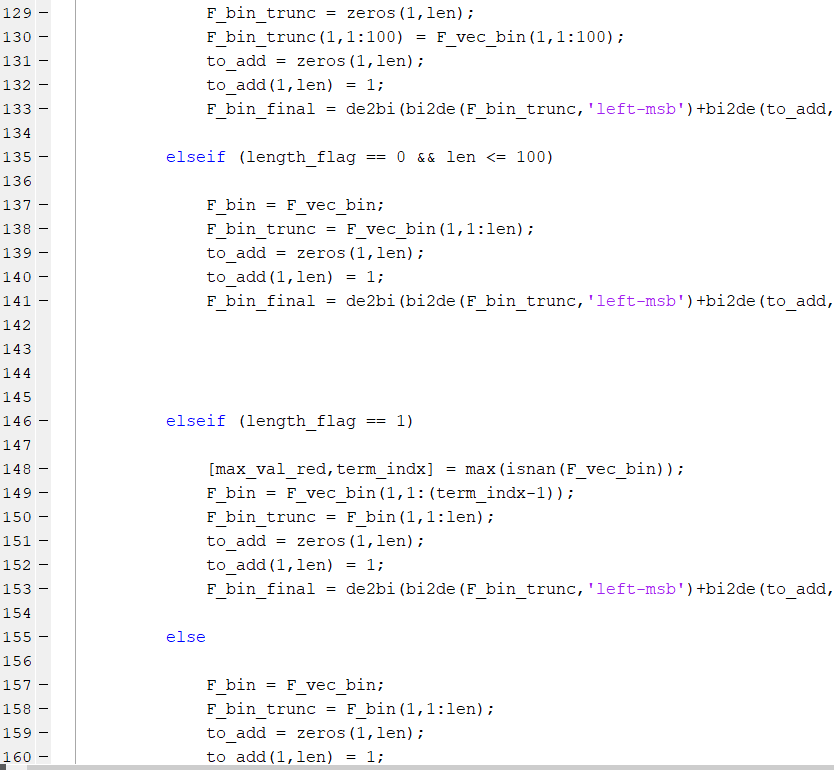
The coding of all results done together. Answer to the question in Part b (the latter part, relating to equations in Moser) and discussion Part c done later

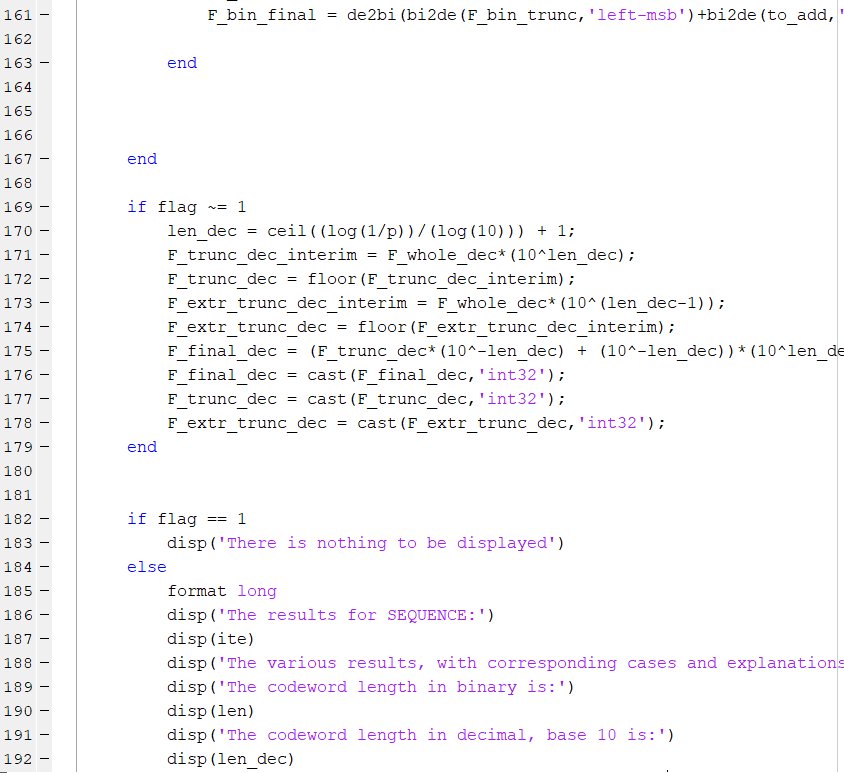


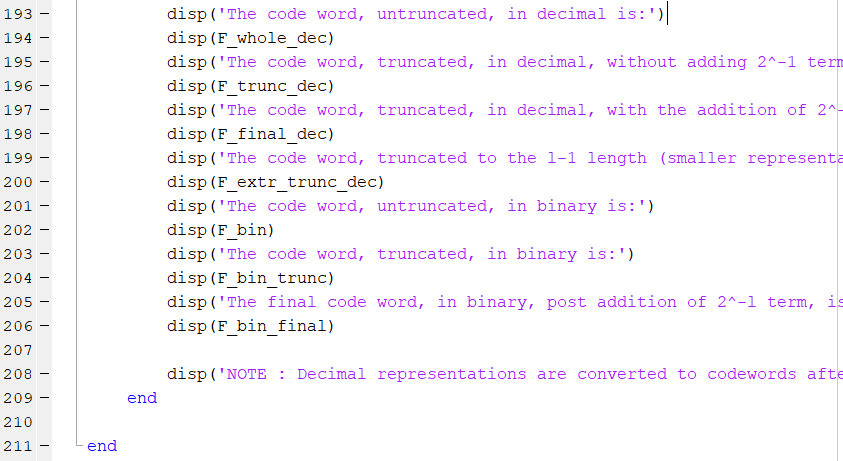




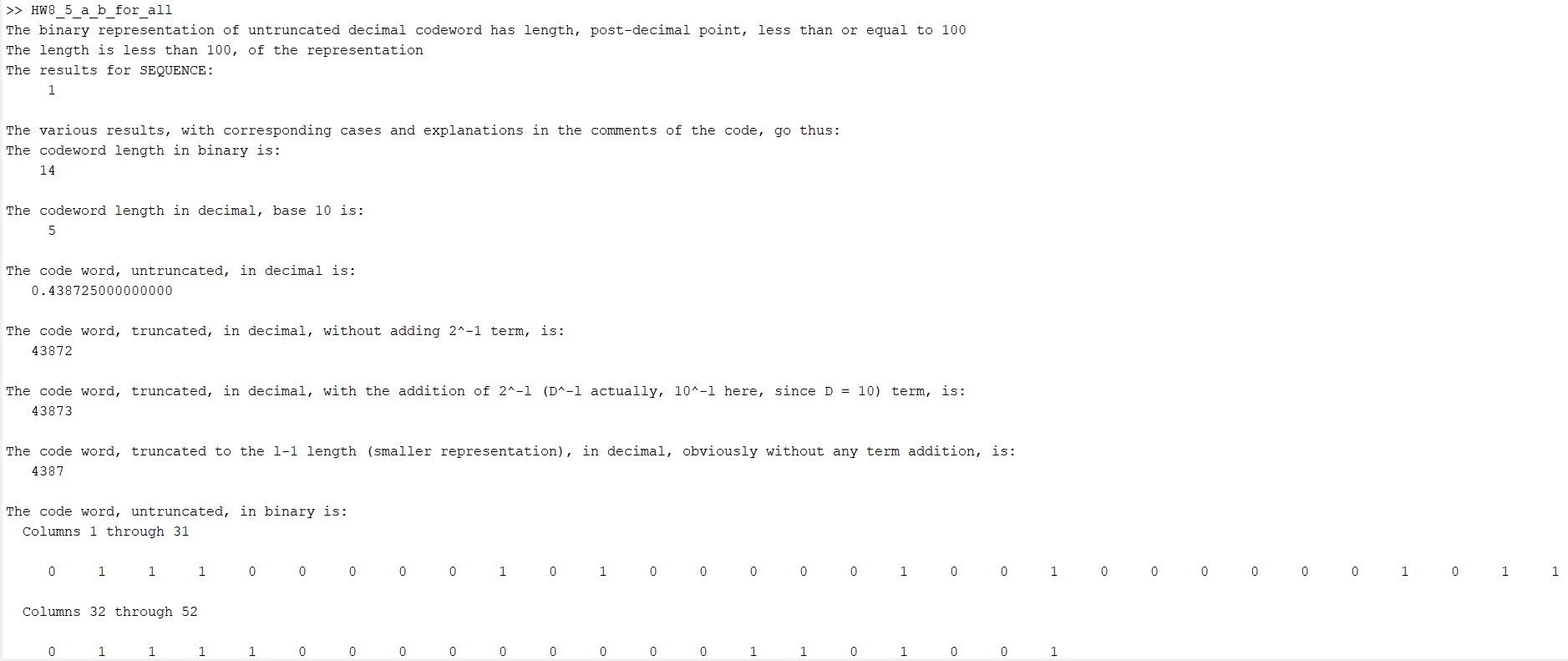


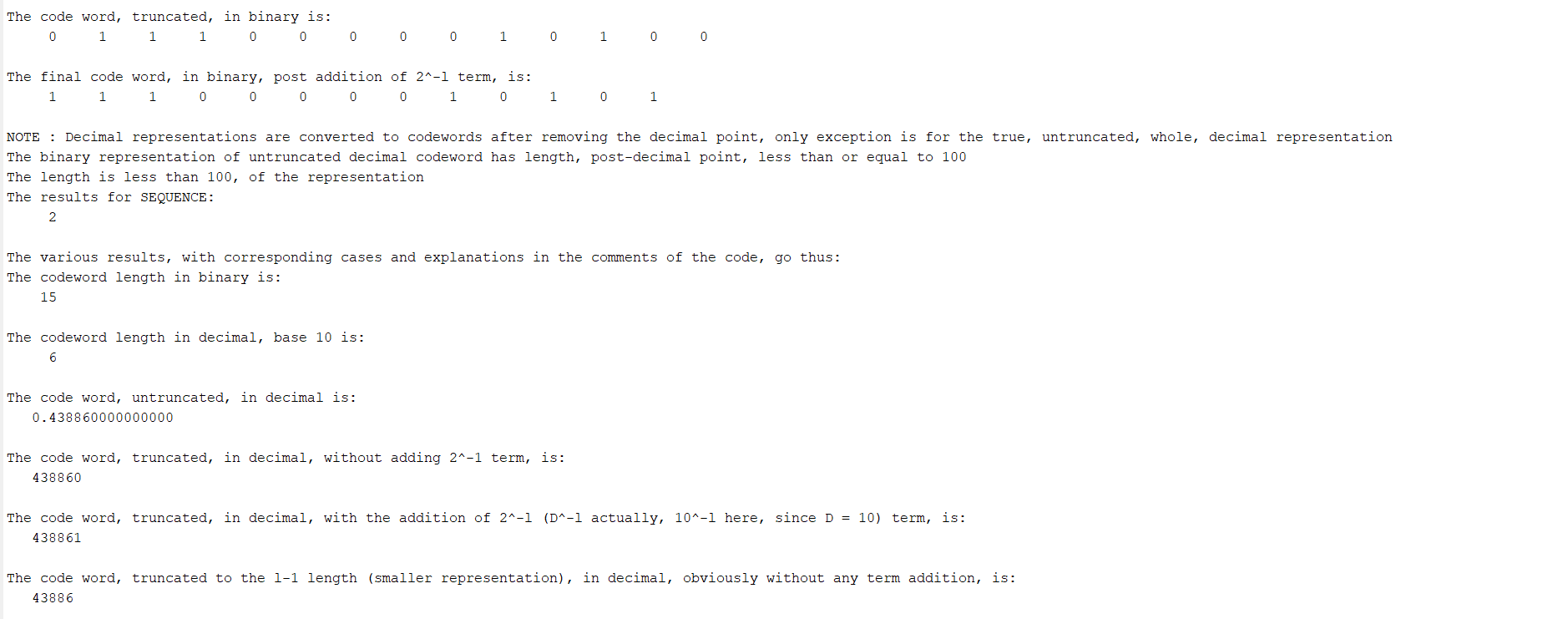


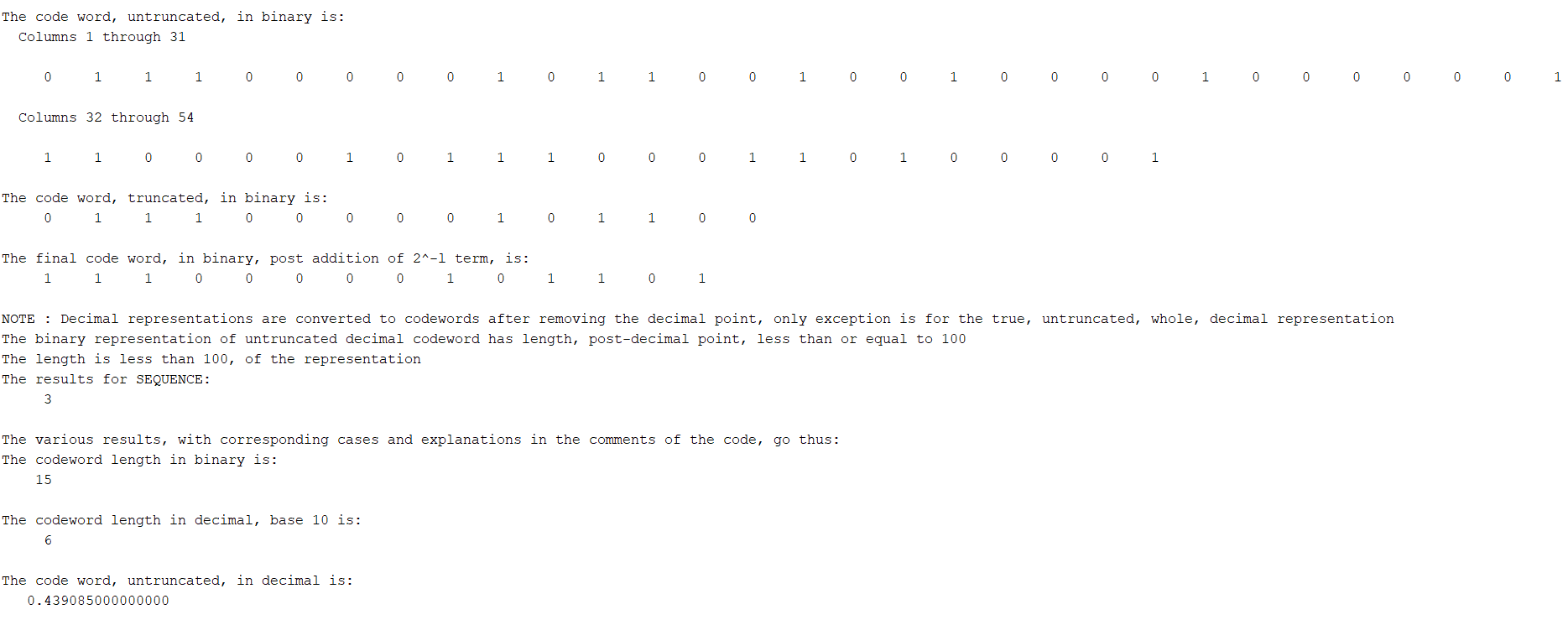


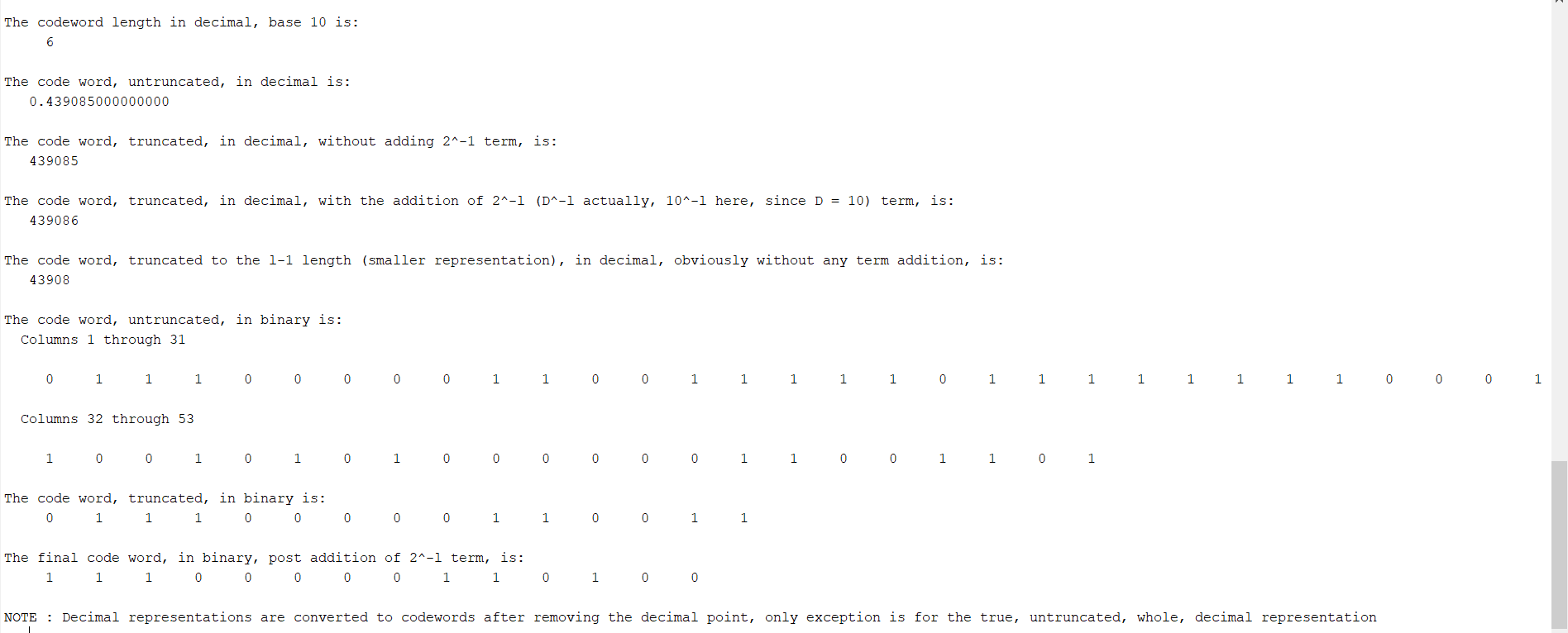


Result Snap









Again, due to the length and width of the results, they can’t be clearly seen in the Result Snaps, so we have copy pasted the entire MATLAB Results Tab to summarize the results

>> HW8\_5\_a\_b\_for\_all

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The results for SEQUENCE:

1

The various results, with corresponding cases and explanations in the comments of the code, go thus:

The codeword length in binary is:

14

The codeword length in decimal, base 10 is:

5

The code word, untruncated, in decimal is:

0.438725000000000

The code word, truncated, in decimal, without adding 2^-1 term, is:

43872

The code word, truncated, in decimal, with the addition of 2^-l (D^-l actually, 10^-l here, since D = 10) term, is:

43873

The code word, truncated to the l-1 length (smaller representation), in decimal, obviously without any term addition, is:

4387

The code word, untruncated, in binary is:

Columns 1 through 31

0 1 1 1 0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 1 0 1 1

Columns 32 through 52

0 1 1 1 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1

The code word, truncated, in binary is:

0 1 1 1 0 0 0 0 0 1 0 1 0 0

The final code word, in binary, post addition of 2^-l term, is:

1 1 1 0 0 0 0 0 1 0 1 0 1

NOTE : Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The results for SEQUENCE:

2

The various results, with corresponding cases and explanations in the comments of the code, go thus:

The codeword length in binary is:

15

The codeword length in decimal, base 10 is:

6

The code word, untruncated, in decimal is:

0.438860000000000

The code word, truncated, in decimal, without adding 2^-1 term, is:

438860

The code word, truncated, in decimal, with the addition of 2^-l (D^-l actually, 10^-l here, since D = 10) term, is:

438861

The code word, truncated to the l-1 length (smaller representation), in decimal, obviously without any term addition, is:

43886

The code word, untruncated, in binary is:

Columns 1 through 31

0 1 1 1 0 0 0 0 0 1 0 1 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1

Columns 32 through 54

1 1 0 0 0 0 1 0 1 1 1 0 0 0 1 1 0 1 0 0 0 0 1

The code word, truncated, in binary is:

0 1 1 1 0 0 0 0 0 1 0 1 1 0 0

The final code word, in binary, post addition of 2^-l term, is:

1 1 1 0 0 0 0 0 1 0 1 1 0 1

NOTE : Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation

The binary representation of untruncated decimal codeword has length, post-decimal point, less than or equal to 100

The length is less than 100, of the representation

The results for SEQUENCE:

3

The various results, with corresponding cases and explanations in the comments of the code, go thus:

The codeword length in binary is:

15

The codeword length in decimal, base 10 is:

6

The code word, untruncated, in decimal is:

0.439085000000000

The code word, truncated, in decimal, without adding 2^-1 term, is:

439085

The code word, truncated, in decimal, with the addition of 2^-l (D^-l actually, 10^-l here, since D = 10) term, is:

439086

The code word, truncated to the l-1 length (smaller representation), in decimal, obviously without any term addition, is:

43908

The code word, untruncated, in binary is:

Columns 1 through 31

0 1 1 1 0 0 0 0 0 1 1 0 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 0 0 1

Columns 32 through 53

1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 1 0 1

The code word, truncated, in binary is:

0 1 1 1 0 0 0 0 0 1 1 0 0 1 1

The final code word, in binary, post addition of 2^-l term, is:

1 1 1 0 0 0 0 0 1 1 0 1 0 0

NOTE : Decimal representations are converted to codewords after removing the decimal point, only exception is for the true, untruncated, whole, decimal representation

>>

The results are all well summarized and described above, including all the details asked and more, and can be referred to. These results are self-explanatory.

**PART b QUESTIONS**

**COMPARISON OF VARIOUS DECIMAL REPRESENTATIONS**

**IMPACT OF 5.23 AND 5.24 MOSER EQUATIONS**

**CAN CODEWORD FALL OUT OF THE DESIRED INTERVAL OF THE CDF?**

Answer:

The reason why 2^-l is added and length made 1 more than the Shannon length is to ensure that the codes remain prefix-free, since, the PMF is not ordered.

The unordering of the PMF corresponds to a situation whereby Arithmetic Coding could give codes that are not prefix-free, if, we used the original Shannon Length and didn’t add 2^-l.

This situation can be also seen from the above sequences and their results, corresponding to representations where we have 2^-l not added to the correctly truncated sequences and where we have truncated the sequences to a length equal to Shannon Length (1 less than normal Arithmetic Coding length).

Since prefix-free nature is no longer guaranteed when we truncate to a shorter length (Shannon Length) or when we don’t add 2^-l to the sequence correctly truncated, the codeword may fall out of the desired cdf interval making the code not prefix-free.

Thus truncating to one more than shannon length and adding 2^-l both ensure that the arithmetic code remains prefix-free, and within correct cdf range.

**PART c**

**RESULT DISCUSSION**

Answer:

We have already discussed things in detail, as such, since the beginning, but in summary, we obtained Arithmetic Codes, both binary and decimal for different versions (truncated, untruncated, incorrectly truncated, correctly truncated without 2^-l added) and we saw how it affects the arithmetic code for different sequences for a ternary random variable.

We can say that truncating to a length more than Shannon Length and adding 2^-l is very important for the Arithmetic Codes, since doing so ensures that code stays prefix-free.