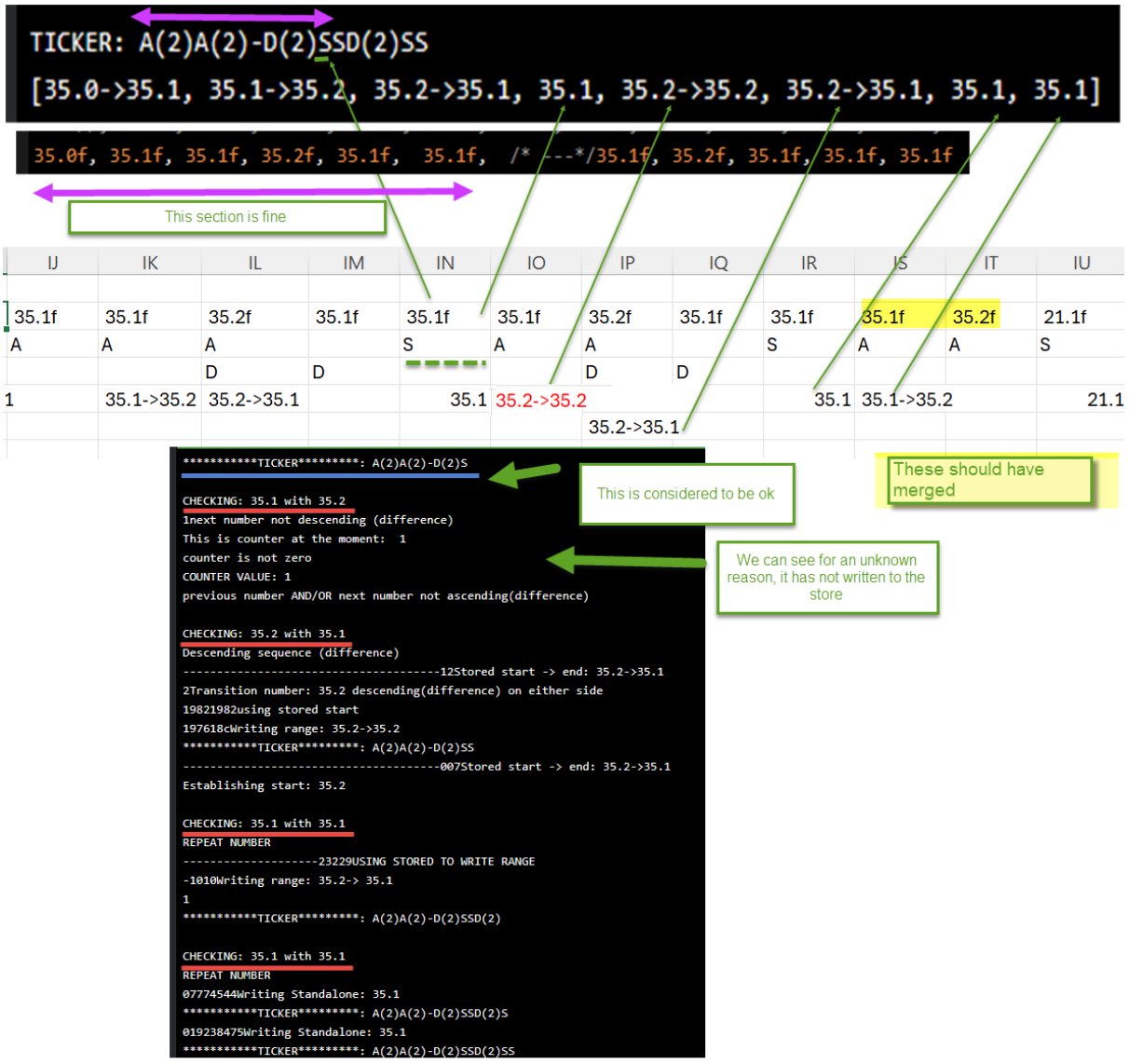
Upon generating a final dataset, these were issues I had to resolve. I am totally unsure impact it will have elsewhere...



I visited this whole section of code again

```
1352                }
1353            }
1354            //the store is empty
1355            else
1356            {
1357                if ((Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
1358                && (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
1359                {
1360                    System.out.println(potentialfurtherAscendingBeyondThisStart);
1361                    System.out.println(potentialfurtherAscendingBeyondThisEnd);
1362                    System.out.println("2bWriting range: " + start + "-> " + end);
1363                    sm.add(start+"->"+end);
1364                    completeTicker(start,end,k,lengthNums);
1365                }
1366
1367                else
1368                {
1369                    //store range here
1370                    if ((Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
1371                    || (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
1372
1373                    start=String.valueOf(nums[k]);
1374                    end=String.valueOf(nums[k+1]);
1375                    potentialfurtherAscendingBeyondThisStart = start;
1376                    potentialfurtherAscendingBeyondThisEnd = end;
1377    System.out.println("-----------------------------------12Stored start -> end: " + potentialfurtherAscendingBeyondThisStart + "->" + potentialfur
1378                }
1379
1380                //I no longer feel this is an else associated with below if
1381                //else
1382                //{
1383                    if ((nums[k]==nums[k-1])
1384                    &&  !(Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
1385                    && !(Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
1386                    {
1387                        start = String.valueOf(nums[k]);
1388                        sm.add(start);
1389                        System.out.println("8888888888Writing Standalone: " + start);
1390                        completeTicker(start, start,k,lengthNums);
1391                    }
1392                //}
1393
1394            }
1395        }
1396        isFirstOccurenceAscendingChain=false;
```

This is the new code added. It would create the store is nums[k+1] is within difference.

I found this logic to be incorrect. It can be seen that I am choosing to write range... But this would only be the case if nums[k+1] is not within difference... I had implemented the logic totally incorrect.. It should have been:

```
if (!(Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
&& !(Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
```

This was previous an else statement, but it made little sense.. So it has become standalone if... I have also narrowed scope massively

```
TICKER: A(2)A(2)-D(2)SA(2)-D(2)SS

[35.0->35.1, 35.1->35.2, 35.2->35.1, 35.1, 35.1->35.2, 35.2->35.1, 35.1, 35.1]

35.0f, 35.1f, 35.1f, 35.2f, 35.1f,  35.1f,  /* ---*/35.1f, 35.2f, 35.1f, 35.1f, 35.1f
```

ALL ISSUES ARE NOW FIXED

I will now focus on the next area in the ChatGPT where there seems to be an error

	KW	KX	KY	KZ	LA	LB	LC	LD
	9.5f	9.4f	9.4f	9.5f	9.6f	9.6f	9.6f	9.7f
	D	D	A	A	A	A	A	A
0.9	9.5->9.4		MISSING	9.5->9.6				

```
0.9f, 0.9f, 0.9f, 0.9f, 9.5f, 9.4f, 9.4f, 9.5f, 9.6f, 9.6f, 9.6f, 9.7f, 60.1f
```

```
TICKER: SSSSD(2)A(2)SA(2)S
[0.9, 0.9, 0.9, 0.9, 9.5->9.4, 9.4->9.6, 9.6, 9.6->9.7, 60.1]
```

It seems this issue is now resolved also due to above fixes

Unfortunately I will need to run the entire 3000 numbers again.
I found a new issue and it all related to the new code I implemented, it needed a bit stringent conditions...

	JZ	KA	KB	KC	KD	KE	KF	KG
	61.5f	61.4f	61.4f	61.4f	22.0f	59.6f	59.6f	59.6f
	D	D	S	S	S	S	S	S
5	61.5->61.4		61.4	61.4	22	59.6	59.6	59.6
	61.5f	61.4f	61.4f	61.4f	22.0f	59.6f	59.6f	59.6f
	D	D	S	S	S	S	S	S
5	61.5->61.4		61.4	61.4->61.4 61.4	22			

```
1.1f, 1.1f, 1.2f, 61.5f, 61.5f, 61.5f, 61.4f, 61.4f, 61.4f, 22.0f, 59.6f, 59.6f
```

```
TICKER: SA(2)SSD(2)SSSSSS
[1.1, 1.1->1.2, 61.5, 61.5, 61.5->61.4, 61.4, 61.4->61.4, 61.4, 22.0, 59.6, 59.6]
```

```
CHECKING: 61.5 with 61.5
REPEAT NUMBER
019238475Writing Standalone: 61.5
**********TICKER*********: SA(2)SS

CHECKING: 61.5 with 61.4
Descending sequence (difference)
-------------------------------------12Stored start -> end: 61.5->61.4
Establishing start: 61.5

CHECKING: 61.4 with 61.4
REPEAT NUMBER
--------------------23229USING STORED TO WRITE RA
-1010Writing range: 61.5-> 61.4
1
**********TICKER*********: SA(2)SSD(2)

CHECKING: 61.4 with 61.4
REPEAT NUMBER
019238475Writing Standalone: 61.4
**********TICKER*********: SA(2)SSD(2)S

CHECKING: 61.4 with 22.0
1next number not descending (difference)
This is counter at the moment:  1
counter is not zero
COUNTER VALUE: 1
previous number AND/OR next number not ascending(difference)
```

This is ok

This is ok

This is ok

From herein, we should be getting two standalone writes of 61.4
But there are two standalone writes and also 61.4->6.14 which is incorrect

This is ok

```
if (k>0)
{
    if (nums[k]==nums[k-1] || nums[k]==nums[k+1])
    {
        sm.add(start);
        System.out.println("019238475Writing Standalone: " + start);
        completeTicker(start, start,k,lengthNums);
        hasWrittenRepeatNumber=true;
    }
}
```

This is part of logic implemented earlier. We can see difference 61.4, 22.0 is larger difference, so it writes the range. I think I should only write range if **hasWrittenRepeatNumber = false...**
Since otherwise it would write incorrect entry as to the right where it has captured start=nums[k-1] and end = nums[k] where both have same values...

```
else
{
    if (!(Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
    && !(Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
    {
        System.out.println(potentialfurtherAscendingBeyondThisStart);
        System.out.println(potentialfurtherAscendingBeyondThisEnd);
        System.out.println("2bWriting range: " + start + "-> " + end);
        sm.add(start+"->"+end);
        completeTicker(start,end,k,lengthNums);
    }
```

```
2bWriting range: 61.4-> 61.4
**********TICKER*********: SA(2)SSD(2)SS
8888888888Writing Standalone: 61.4
**********TICKER*********: SA(2)SSD(2)SSS

CHECKING: 22.0 with 59.6
1next number not descending (difference)
This is counter at the moment:  0
2next number not descending (difference)
previous number ascending (difference) OR next number descending(difference)
6Writing Standalone: 22.0
**********TICKER*********: SA(2)SSD(2)SSSS

CHECKING: 59.6 with 59.6
REPEAT NUMBER
07774544Writing Standalone: 59.6
**********TICKER*********: SA(2)SSD(2)SSSSS
019238475Writing Standalone: 59.6
**********TICKER*********: SA(2)SSD(2)SSSSSS
**********TICKER*********: SA(2)SSD(2)SSSSSS

TICKER: SA(2)SSD(2)SSSSSS
[1.1, 1.1->1.2, 61.5, 61.5, 61.5->61.4, 61.4, 61.4->61.4, 61.4, 22.0, 59.6, 59.6]
Standalone numbers: 9   Ascending chains: 1   Descending chains: 1      TOTAL: 11
Transition events: 0
```

Logic is flawed here

This is also part of the code that I adjusted earlier. I think this part of the code can be run untouched given above modification

```
//I no longer feel this is an else associated with below if
//else
//{
    if ((nums[k]==nums[k-1])
    &&  !(Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
    && !(Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
    {
        start = String.valueOf(nums[k]);
        sm.add(start);
        System.out.println("8888888888Writing Standalone: " + start);
        completeTicker(start, start,k,lengthNums);
    }
```

```
[1.1, 1.1->1.2, 61.5, 61.5, 61.5->61.4, 61.4, 61.4, 22.0, 59.6, 59.6]
```

It appears that it has completed the ChatGPT extract 3 successfully.
However I was extremely aware that dataset did not include two consecutive transition elements such as Ascending, Descending, Ascending

So I am going through all my test cases and found an issue as follows

I am still quite doubtful on all my test cases when it reaches k==nums.length-2

Once I have finished test cases, I will devise a few more in which it has to make various decisions at this point based on previous numbers...

At the moment, I have found a failed test case as below:

```
CHECKING: 4.7 with 4.8
1next number not descending (difference)
This is counter at the moment:  0
next number ascending (difference)
-------------------------------------0Stored start -> end: 4.7->4.8

CHECKING: 4.8 with 4.9
1next number not descending (difference)
This is counter at the moment:  0
2next number not descending (difference)
previous number descending(difference) AND/OR next number ascending (difference)
Next number ascending
Previous number descending (difference) AND next number ascending (difference)

CHECKING: 4.9 with 5.0
1next number not descending (difference)
This is counter at the moment:  0
2next number not descending (difference)
previous number descending(difference) AND/OR next number ascending (difference)
Next number ascending
Previous number descending (difference) AND next number ascending (difference)

CHECKING: 5.0 with 5.0
REPEAT NUMBER
--------------------23229USING STORED TO WRITE RANGE
-1010Writing range: 4.7-> 5.0

CHECKING: 5.0 with 4.9
Descending sequence (difference)
```

As I examined this code, I could see I set the value hasWrittenRepeatNumber=true

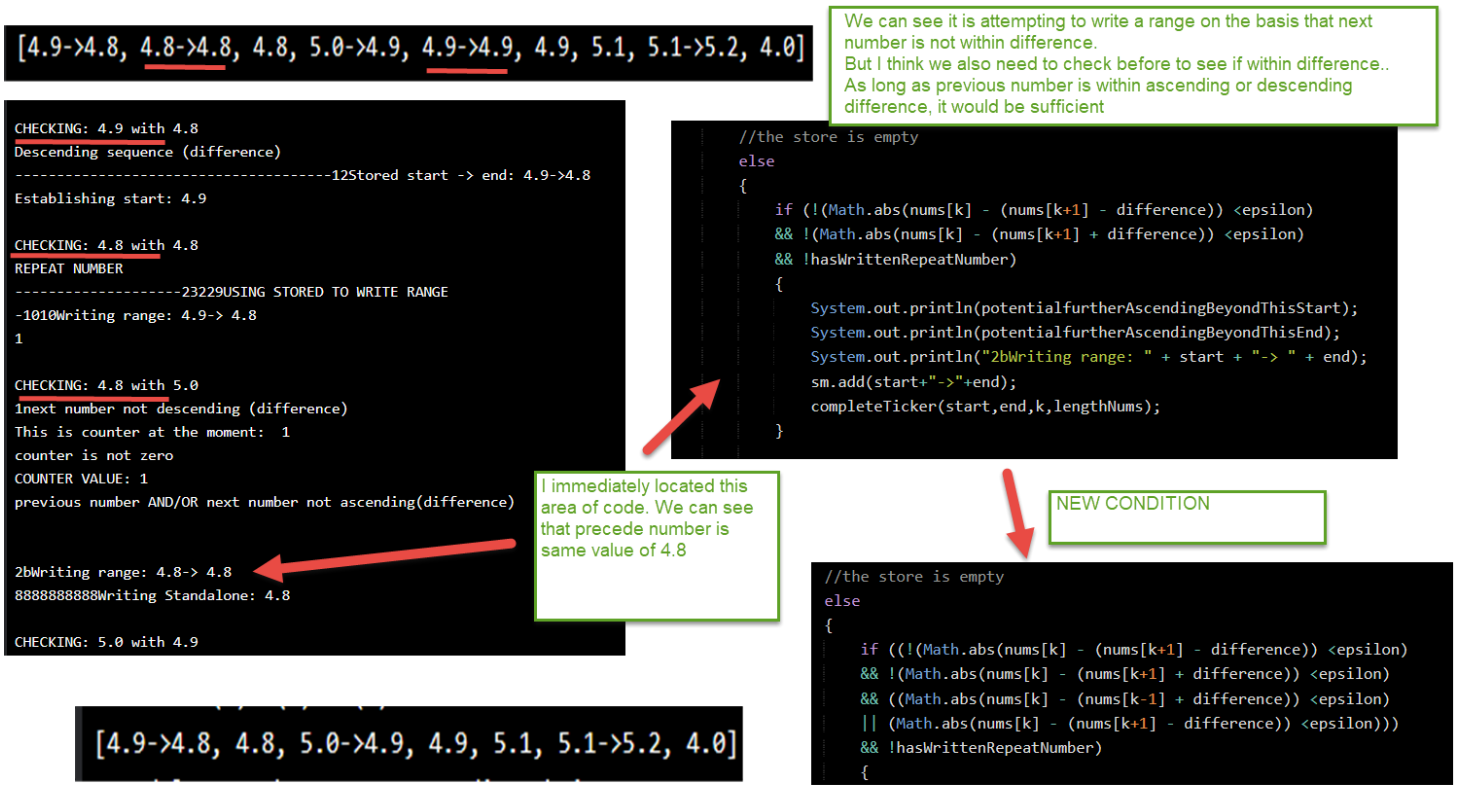
But this area of code is right at the bottom.. And it has not been set back to false.
From understanding of my code, I set these booleans to enforce mutual exclusiveness during a single iteration of the k loop.
So I think commenting this out is a good decision.

```
1497                if((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
1498                || (Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
1499                {
1500                    if ((Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
1501                    {
1502                        ascendingCounter++;
1503                    }
1504
1505                    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
1506                    System.out.println("--------------------23229USING STORED TO WRITE RANGE");
1507                    System.out.println("-1010Writing range: " + potentialfurtherAscendingBeyondThisS
1508                    completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),
1509                    //hasWrittenRepeatNumber=true;  //late change
1510
1511                    potentialfurtherAscendingBeyondThisStart="";
1512                    potentialfurtherAscendingBeyondThisEnd="";
1513
1514                    if (String.valueOf(nums[k+1]).equals(potentialfurtherAscendingBeyondThisEnd))
1515                    {
1516                        sm.add(start);
```

```
TICKER: A(4)D(2)
[4.7->5.0, 5.0->4.9]
```

I consider this to be a highly critical change and I will run through test cases again from the top..

I found an issue with this test case



I am finding that all issues occurring are related to the new code I introduced in this document. But I seem to be resolving them...

Unfortunately I discovered one of my changes above was detrimental, but I found a solid workaround... It is now a case of undoing a change (see overleaf).

Overleaf is the code that has been introduced:

```
sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
System.out.println("--------------------23229USING STORED TO WRITE RANGE");
System.out.println("-1010Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nu
completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
hasWrittenRepeatNumber=true;
tickerCounterOnPreviousRange=tickerCounter;
```

I have kept a copy of the tickerCounter

I introduced this new code in the k=nums.length-2 section. Basically when it reaches here and it is ready to write the range (on basis that no others have been written since the above). It also is more failproof acknowledging the tickerCounter>0 to ensure it can not enter here inadvertently... It will write 50.0-<49.9

```
665                 if ((potentialfurtherAscendingBeyondThisStart=="")
666                 && (potentialfurtherAscendingBeyondThisEnd=="")
667                 && (hasWrittenRepeatNumber)
668                 && (tickerCounter==tickerCounterOnPreviousRange)
669                 && tickerCounter>0)
670                 {
671                     start=String.valueOf(nums[k]);
672                     end=String.valueOf(nums[k+1]);
673                     System.out.println("1201021Writing range: " + start + "-> " + end);
674                     sm.add(start+"->" + end);
675                     completeTicker(start,end,k,lengthNums);
676                     tickerCounterOnPreviousRange=0;
677                     break;
678                 }
```

```
[4.7->5.0, 5.0->4.9]
```

I will now try ChatGPT extract 3 again, all seems ok

Before I go through all my test cases again, I am quite keen to revisit those test cases that failed in last documentation due to A combinations of A with A-D....

I will see if they still exist, note these are ONLY issues with the ticker.

//small extract taken from second chatGPT extract. ok, ticker=fail (USE EXAMPLE)

//40.1f, 40.1f, 35.1f, 35.2f, 35.3f, 35.2f, 35.1f, 85.6f, 85.5f, 85.4f, 85.3f, 85.2f,19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f

//3.7f,3.8f,3.7f,3.9f,4.0f,3.5f,3.6f,3.7f,3.8f,3.7f,45.5f,45.4f,3f,3.2f,3.3f //ok, ticker=fail A (A-D)

//ISSUES THESE //Related to oversized or undersized ticker, it relates to transition and also standalone value occuring prior to it

//3.5f,3.6f,3.7f,3.8f, 3.7f, 3.5f, 3.6f,3.7f,3.8f, 3.7f,3.6f, 3.5f,3.6f,3.7f,3.8f //ok, ticker=fail (A-D) (A-D) A

//5.0f, 48.5f,28.6f, 28.5f, 28.4f, 28.3f, 28.2f, 3.5f,3.6f, 3.5f //ok, ticker=ok //issues D (A - too large) D

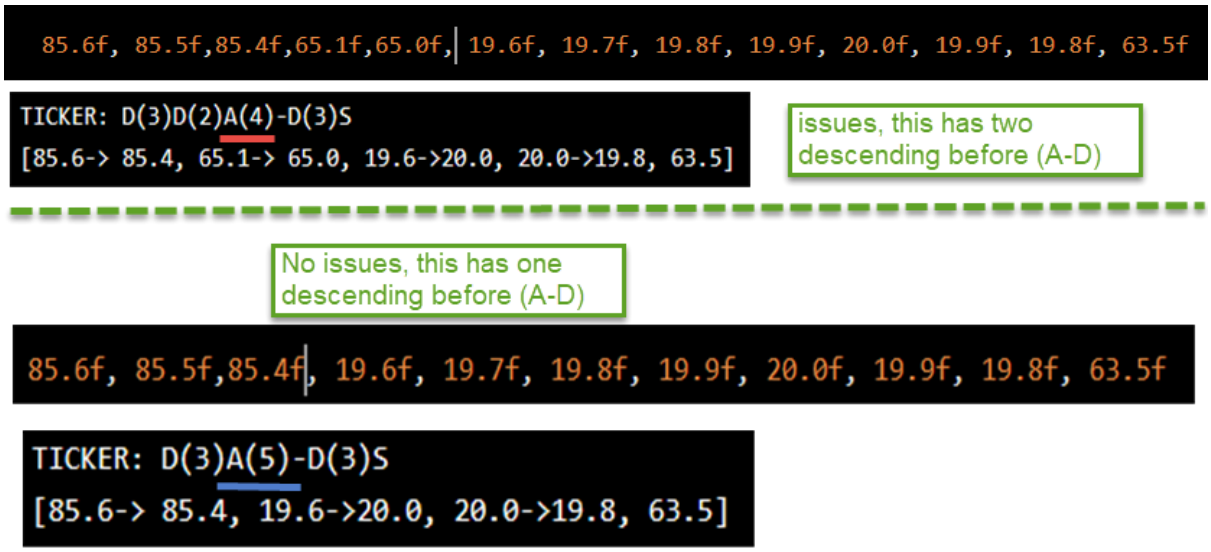
//28.6f, 28.5f, 28.4f, 28.3f, 28.2f, 3.5f,3.6f, 3.5f //rule does not work DAD //issues ok, ticker=ok

//It appears that ascendingcounter SHOULD NOT BE INCREASED IN THE NEW RULE IN THE COMPLETETICKER METHOD WHERE

//ascendingcounter==1 and descending in both directions....

//29.9f,30.0f,28.9f,28.8f,28.9f,29.0f //no issues ADA //ok, ticker=ok

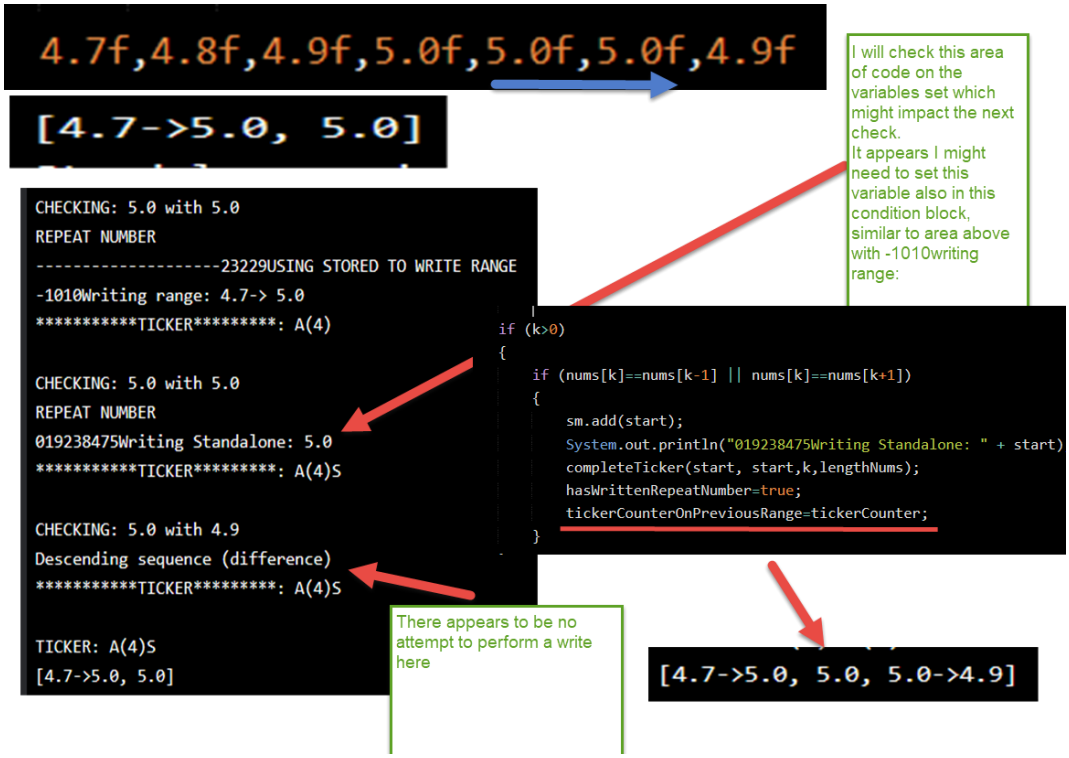
I am now observing ticker being incorrect for a situation not experienced before.



I will aim for now to discard these and try to acknowledge failed ticker whilst I go through test cases....

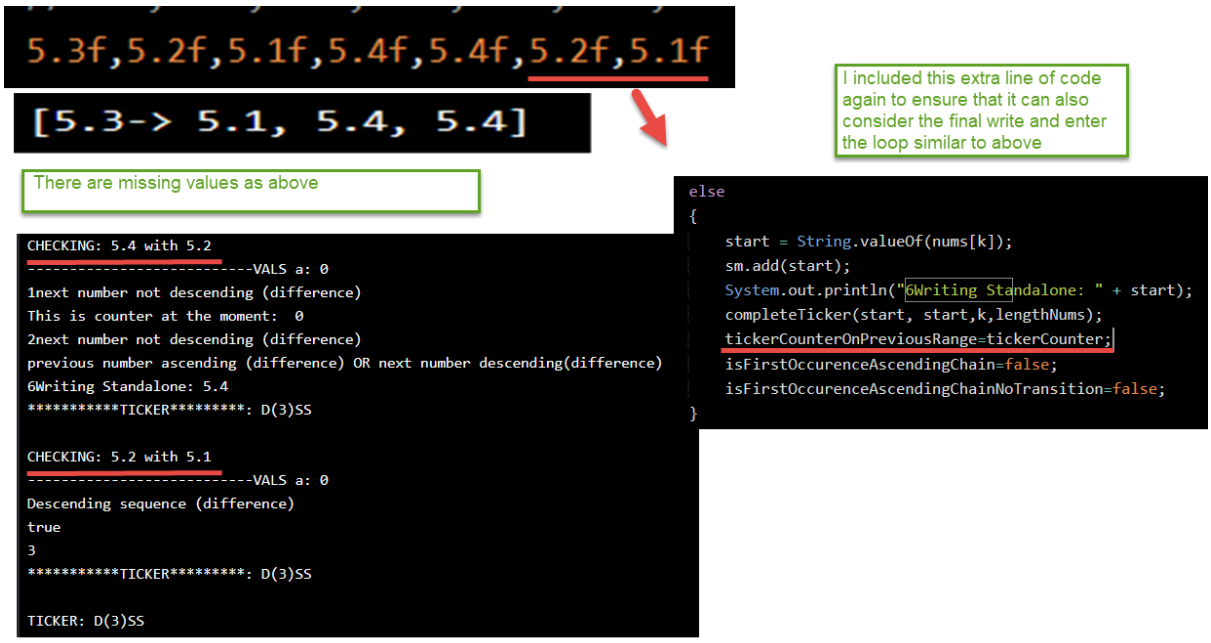
I might not be able to identify all of them given long test arrays..

I found another test case failing, it is similar concept to the above with new variable:



This is another failed case which I remediated

As I suspected earlier, all the scenarios are related to dealing with k=nums.length-2



The only other scenarios are having descending summary range and ascending summary range before it, and there are no issues... I also tried repeat numbers which match with the descent starting at k=nums.length-2 and no issues.
I also changed the last range to ascending and no issues.