I am now trying once more from my final test documentation 30032025/SummaryRange/9/Repository/Solution-Best attempt (fail inline with document 2).java
and the latest failed test code in the repository to try and resolve this issue.

```
4.9f,5.0f,5.1f,5.1f,5.2f,4.0f //standalone is not at k==nums.length-2
```

```
[4.9->5.1, 5.1->5.2, 4.0]
```

I am now going to include a test case with earlier repeat numbers.

```
Next number ascending
First occurrence three consecutive ascending numbers (difference)
-----------------------------------22Stored start -> end: 5.0->5.1
REPEAT
CHECKING: 5.1with 5.1
CURRENTLY IN LIST: [4.9->4.8]
5.1
5.1
5.0
REACH HERE !!!!!!!!
-------------------23229USING STORED TO WRITE RANGE
10101010Writing range: 5.0-> 5.1
asc counter: 1
desc counter: 0

CHECKING: 5.1 with 5.2


currently in list: [4.9->4.8, 5.0->5.1]
```

We can see it has not written the standalone

Perhaps I need query to see if the following two above are equal..

nums[k+1] && potentialfurtherAscendingBeyondThisEnd

```
4.9f,4.8f,4.8f,5.0f,5.1f,5.1f,5.2f,4.0f
```

```java
        //with previous and not next since next it repeat number
        ((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
        || (Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
        {

            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
        System.out.println("-------------------23229USING STORED TO WRITE RANGE");
        System.out.println("10101010Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
        completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
        potentialfurtherAscendingBeyondThisStart="";
        potentialfurtherAscendingBeyondThisEnd="";
        }
        else  //the previous number is too wide and need to write it as a standalone
        {
            System.out.println("INSIDE HERE!!!!");
            standaloneTemp = start;
        }
```

We need to basically separate this code here as below:

```java
        if
        //if previous number within difference, we need to write the summary
        //with previous and not next since next it repeat number
        ((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
        || (Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
        {
            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
        System.out.println("-------------------23229USING STORED TO WRITE RANGE");
        System.out.println("10101010Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
        completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);

         if (String.valueOf(nums[k+1]).equals(potentialfurtherAscendingBeyondThisEnd))
            {
                System.out.println("THIS SECTION$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$4");
                sm.add(start);
                System.out.println("0690690695Writing Standalone: " + start);
                completeTicker(start, start,k,lengthNums);
            }
        potentialfurtherAscendingBeyondThisStart="";
        potentialfurtherAscendingBeyondThisEnd="";
        }
```

NEW CODE

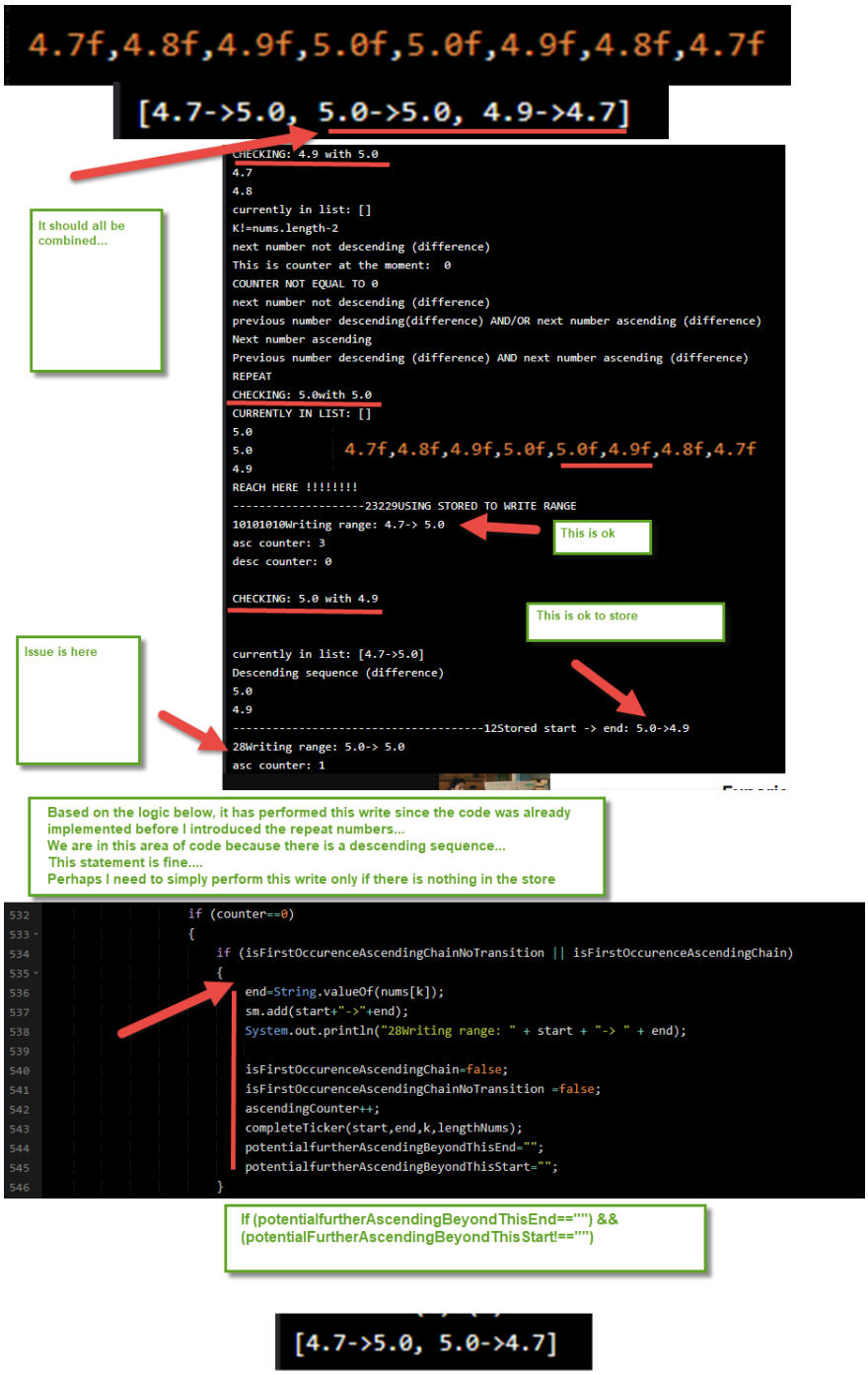
```
4.9f,4.8f,4.8f,5.0f,5.1f,5.1f,5.2f,4.0f
```

```
[4.9->4.8, 5.0->5.1, 5.1->5.2, 4.0]
```

I need to now be extensive in this criteria...

I can think of most basic scenarios as follows
NOTE: k=nums.length-2 refers to number two digits from the end

Ascending (with repeat numbers in range, not part of k=nums.length-2), descending
4.7f,4.8f,4.9f,5.0f,5.0f,4.9f,4.8f,4.7f



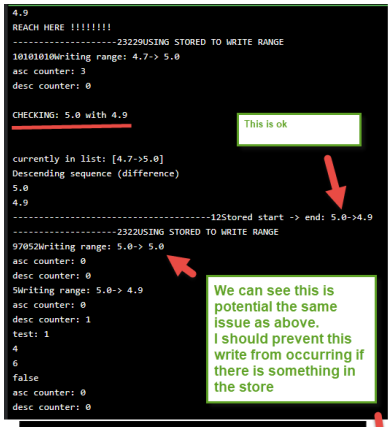
4.7f,4.8f,4.9f,5.0f,5.0f,5.0f,4.9f,4.8f,4.7f

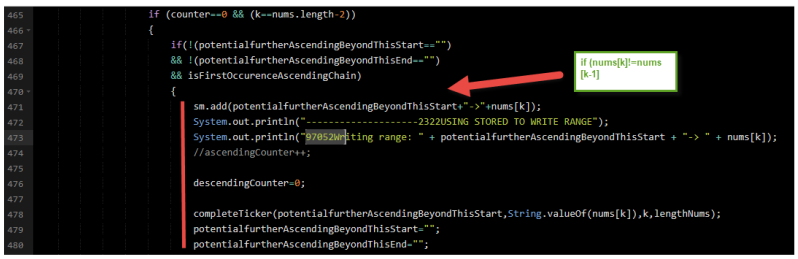
Ascending (with repeat numbers in range, not part of k=nums.length-2 but as part of transition), descending

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

Ascending (with repeat numbers in range, part of k=nums.length-2), descending
4.7f,4.8f,4.9f,5.0f,5.0f,4.9f
Incorrect

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



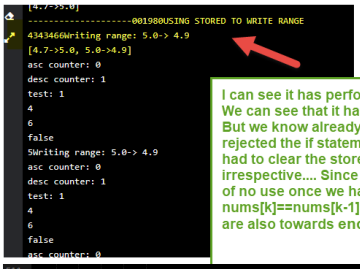



```
4.7f,4.8f,4.9f,5.0f,5.0f,4.9f
[4.7->5.0, 5.0->5.0, 5.0->4.9]
```

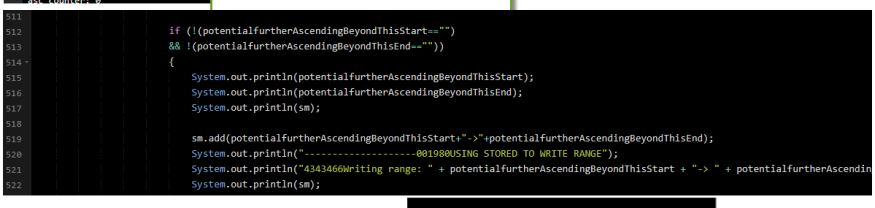
However if we see the loop below, it is in total violation..... and will ruin the logic here...

I can see output is still wrong

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




```
4.7f,4.8f,4.9f,5.0f,5.0f,4.9f
```



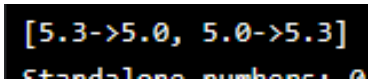

```
[4.7->5.0, 5.0->4.9]
```

Ascending (with more repeat numbers in range, part of k=nums.length-2), descending
4.7f,4.8f,4.9f,5.0f,5.0f,5.0f,4.9f

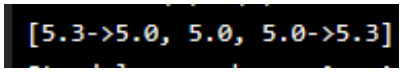
FOR NOW, I AM ALSO NOTICING THAT IT IS FALSELY REPORTING THE SIZE OF DESCEND AND ASCEND... I CAN ONLY FORSEE THAT I REQUIRE AN INCREASE IN COUNTERS IN CERTAIN AREAS... I WILL DEAL WITH THIS LATER. MY MAIN PRIORITY IS ADDING CORRECT CONTENT INTO THE LIST

I will now try the above but reverse the ascending with descending ....

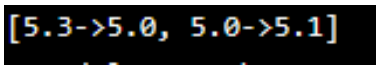
5.3f,5.2f,5.1f,5.0f,5.0f,5.1f,5.2f,5.3f



5.3f,5.2f,5.1f,5.0f,5.0f,5.0f,5.1f,5.2f,5.3f



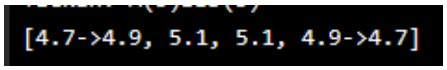
5.3f,5.2f,5.1f,5.0f,5.0f,5.1f



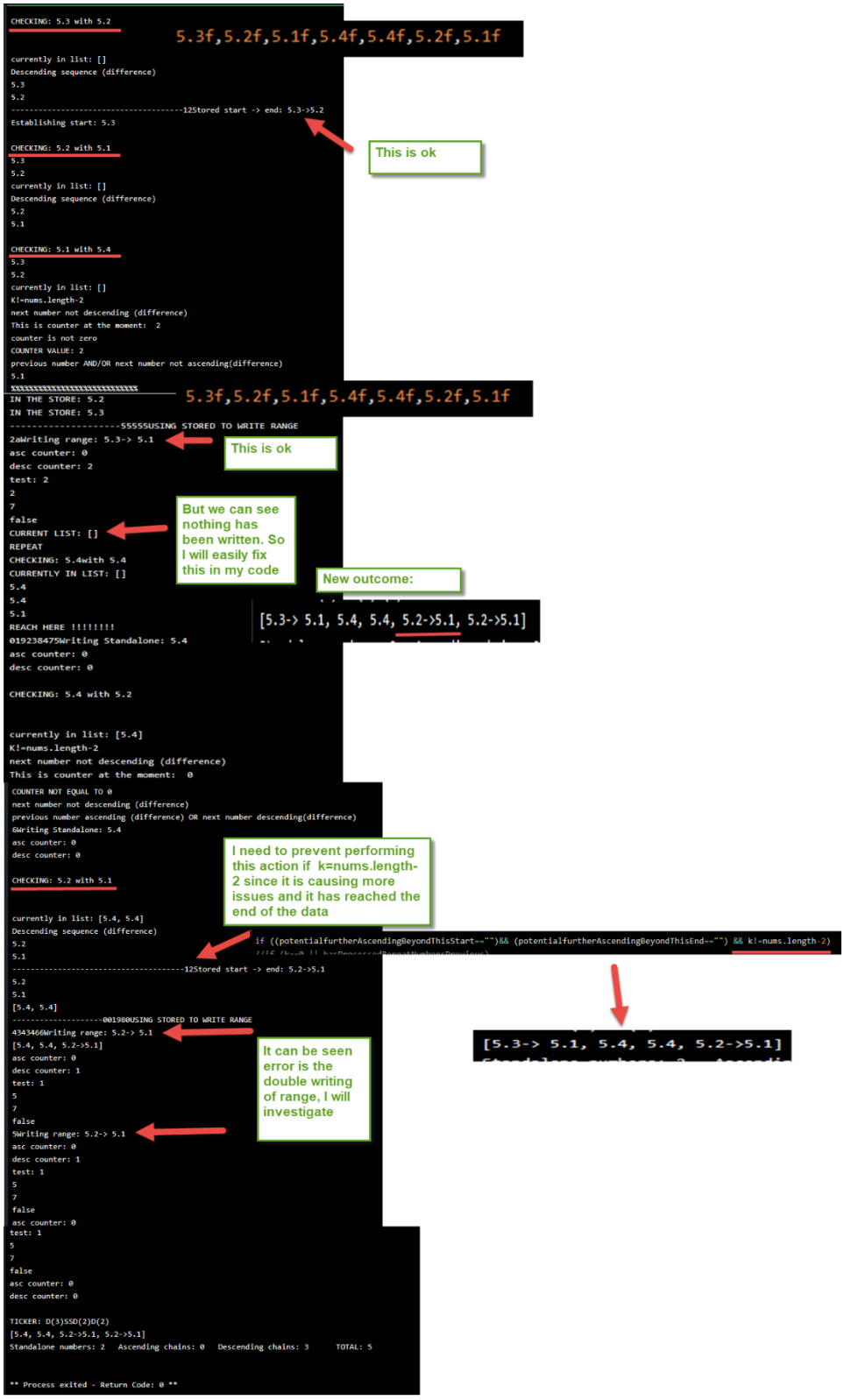
5.3f,5.2f,5.1f,5.0f,5.0f,5.0f,5.1f

I will now need to re-check to ensure it functions in some of the above cases, but this time the standalone are **not** part of the summary range...

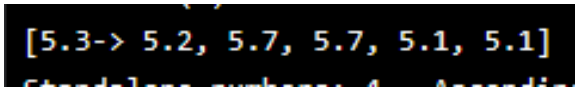
4.7f,4.8f,4.9f,5.1f,5.1f,4.9f,4.8f,4.7f



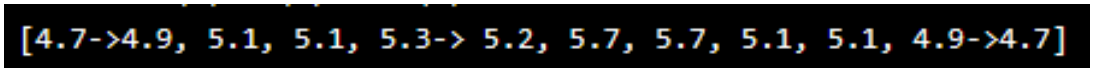
5.3f,5.2f,5.1f,5.4f,5.4f,5.2f,5.1f



5.3f,5.2f,5.7f,5.7f,5.1f,5.1f

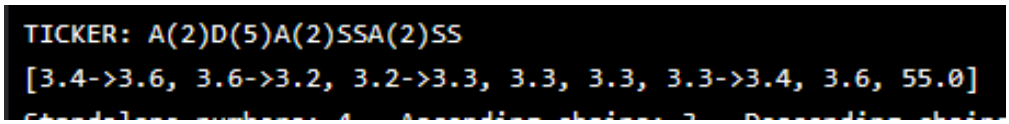


4.7f,4.8f,4.9f,5.1f,5.1f,5.3f,5.2f,5.7f,5.7f,5.1f,5.1f,4.9f,4.8f,4.7f

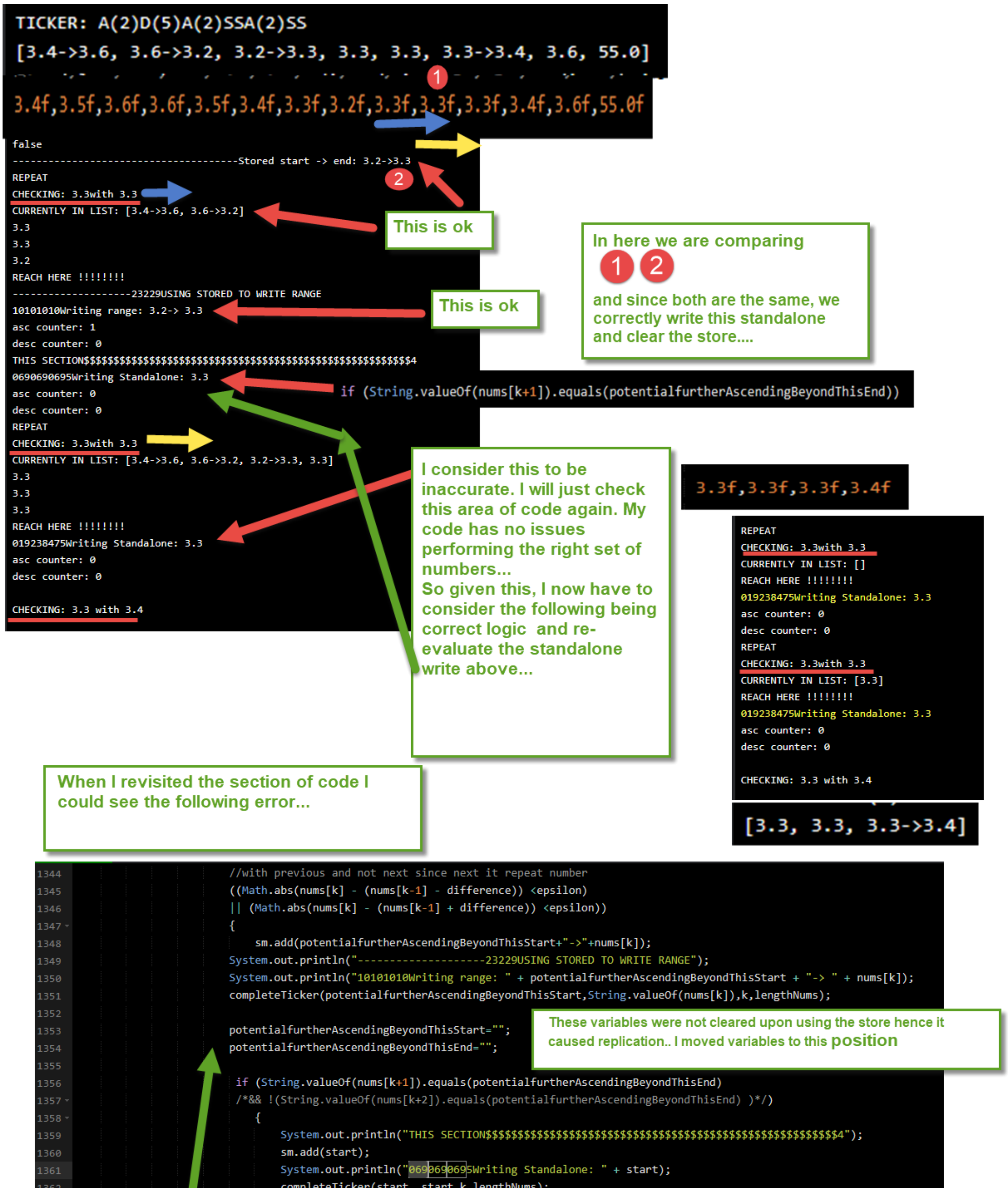


It looks as if I have tackled the last pending issue.
But I still believe I need to find more cases to experiment with..
Perhaps I should aim for:
ascending (repeat numbers for transition) descending (repeat numbers transition) ascending

3.4f,3.5f,3.6f, 3.6f,3.5f,3.4f,3.3f,3.2f, 3.3f, 3.3f,3.3f,3.4f,3.6f,55.0f



I can see it has written 3.3 too many times...



With such a critical change, I will go through all my test cases again in this document

```
4.9f,4.8f,4.8f,5.0f,4.9f,4.9f,5.1f,5.1f,5.2f,4.0f
[4.9->4.8, 5.0->4.9, 5.1, 5.1->5.2, 4.0]
```

I am unsure why this is the only failing test case

```
CHECKING: 4.9 with 4.8

currently in list: []
Descending sequence (difference)
4.9
4.8
-------------------------------------12Stored start -> end: 4.9->4.8
Establishing start: 4.9
REPEAT
CHECKING: 4.8with 4.8
CURRENTLY IN LIST: []
REACH HERE !!!!!!!!
--------------------23229USING STORED TO WRITE RANGE
10101010Writing range: 4.9-> 4.8
asc counter: 0
desc counter: 1
test: 1
1
10
false

CHECKING: 4.8 with 5.0


currently in list: [4.9->4.8]
K!=nums.length-2
next 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)

CHECKING: 5.0 with 4.9


currently in list: [4.9->4.8]
Descending sequence (difference)
5.0
4.9
-------------------------------------12Stored start -> end: 5.0->4.9
Establishing start: 5.0
REPEAT
CHECKING: 4.9with 4.9
CURRENTLY IN LIST: [4.9->4.8]
REACH HERE !!!!!!!!
--------------------23229USING STORED TO WRITE RANGE
10101010Writing range: 5.0-> 4.9
asc counter: 0
desc counter: 1
test: 1
4
10
false
```

This is ok

This is ok

This is ok

It has no effort to write standalone and moved onto next check

I will use this area to make my observations

```
1243                    //if store is empty.
1244                    //we need to only create a summary if the next number is within range
1245
1246                    else
1247                    {
1248                         if ((Math.abs(nums[k] - (nums[k+1] - difference)) <epsilon)
1249                        && (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon))
1250                        {
1251                       System.out.println(potentialfurtherAscendingBeyondThisStart);
1252                       System.out.println(potentialfurtherAscendingBeyondThisEnd);
1253                       System.out.println("2bWriting range: " + start + "-> " + end);
1254                       sm.add(start+"->"+end);
1255                       completeTicker(start,end,k,lengthNums);
1256                        }
```

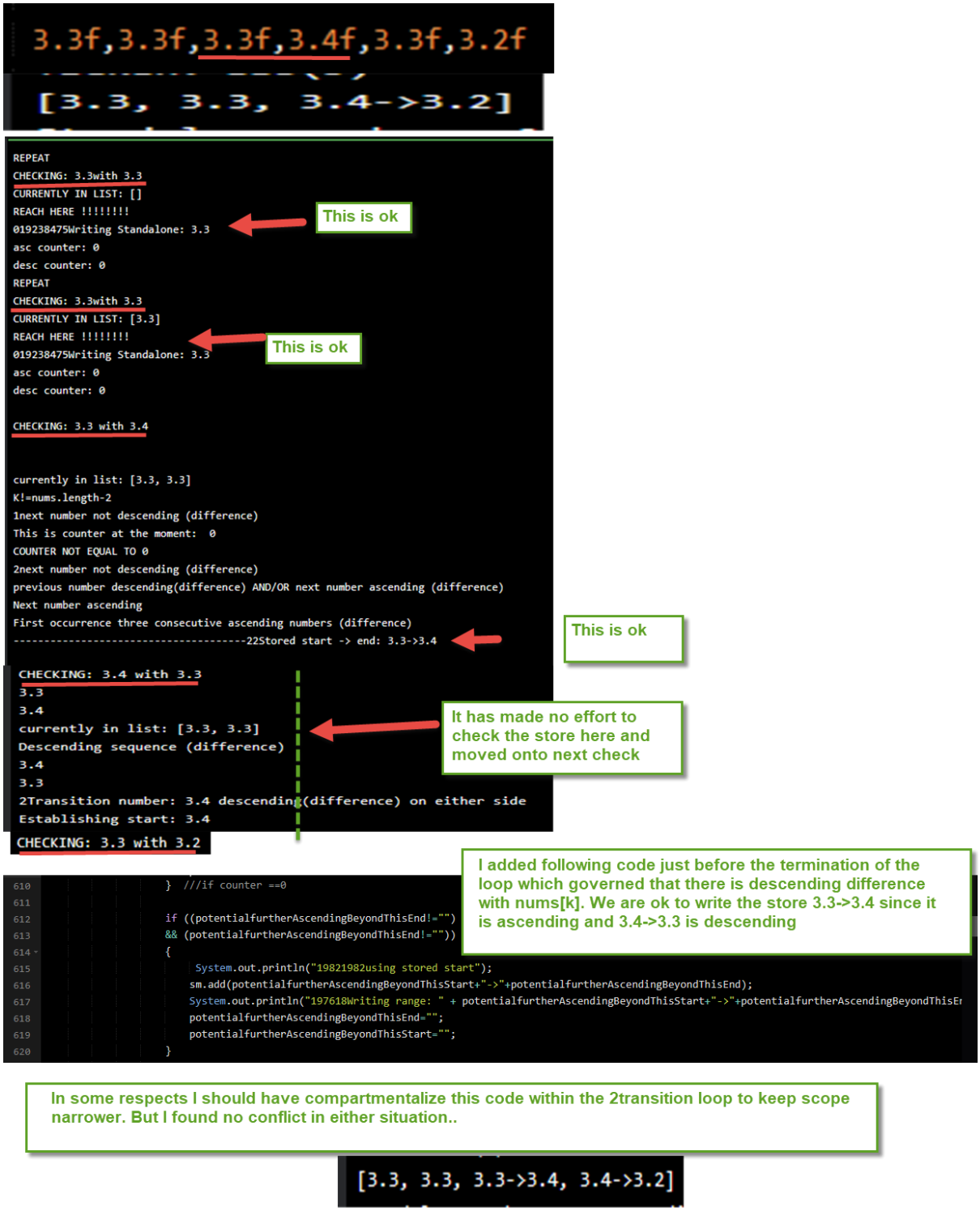
It seems it will not enter here because this checks if next number is part of sequence. (4.8f, 5.0f)
I believe I will need an associated else statement to check if previous number is same. If so, it would add standalone for nums[k]

```
1257                    else
1258                    {
1259                        if (nums[k]==nums[k-1])
1260                        {
1261                            start = String.valueOf(nums[k]);
1262                            sm.add(start);
1263                            System.out.println("8888888888Writing Standalone: " + start);
1264                            completeTicker(start, start,k,lengthNums);
1265                        }
1266                    }
```

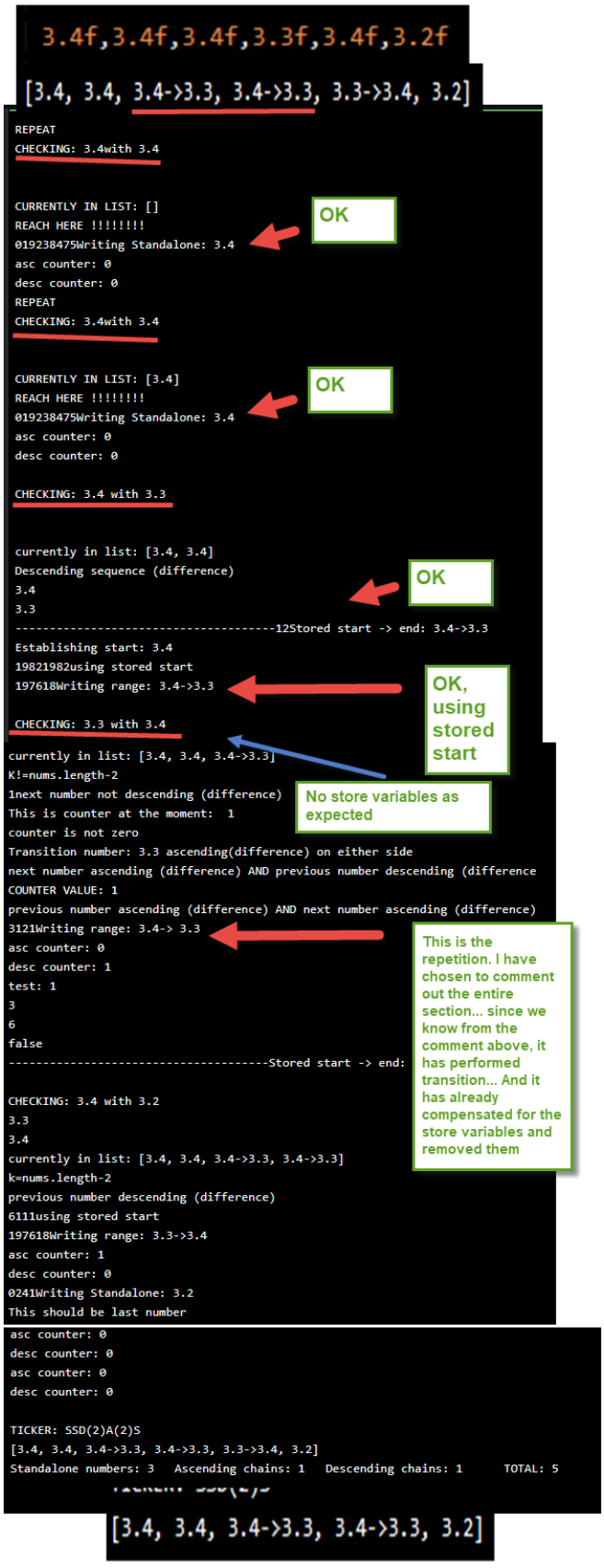
```
[4.9->4.8, 4.8, 5.0->4.9, 4.9, 5.1, 5.1->5.2, 4.0]
```

I will now check all my devised test cases again.
However it just seems that there are always new test cases I have not explored....
But at moment, all the test cases are passing...

So I now have to try all remaining test cases in my code.



I will now reverse the numbers for this test condition.



I do not think I can generate many more cases.. I am just going to combine a few of them into one and see outcome...

```
3.4f,3.4f,3.3f,3.3f,3.3f,3.4f
```

```
[3.4, 3.4->3.3, 3.3, 3.3, 3.3->3.3, 3.3->3.4]
```

```
REPEAT
CHECKING: 3.4with 3.4

CURRENTLY IN LIST: []
REACH HERE !!!!!!!
019238475Writing Standalone: 3.4
asc counter: 0
desc counter: 0

CHECKING: 3.4 with 3.3

currently in list: [3.4]
Descending sequence (difference)
3.4
3.3
-------------------------------------12Stored start -> end: 3.4->3.3
Establishing start: 3.4
19821982using stored start
197618Writing range: 3.4->3.3
REPEAT
CHECKING: 3.3with 3.3

CURRENTLY IN LIST: [3.4, 3.4->3.3]
REACH HERE !!!!!!!
019238475Writing Standalone: 3.3
asc counter: 0
desc counter: 1
REPEAT
CHECKING: 3.3with 3.3

CURRENTLY IN LIST: [3.4, 3.4->3.3, 3.3]
REACH HERE !!!!!!!
019238475Writing Standalone: 3.3
asc counter: 0
desc counter: 1

CHECKING: 3.3 with 3.4

currently in list: [3.4, 3.4->3.3, 3.3, 3.3]
k=nums.length-2
3mWriting range: 3.3-> 3.3
asc counter: 0
desc counter: 1
ascending/descending next number (diffference)
49Writing range: 3.3-> 3.4
asc counter: 0
desc counter: 1
asc counter: 0
desc counter: 1

TICKER: SSSSA(2)
[3.4, 3.4->3.3, 3.3, 3.3, 3.3->3.3, 3.3->3.4]
Standalone numbers: 4   Ascending chains: 1   Descending chains: 0      TOTAL: 5
```

OK

OK

OK

OK

Logic not correct here

We know this is correct since

```
1353            if (!isPrevNumAscending) //careful
1354            {   //new code
1355                System.out.println("REACH HERE !!!!!!!!");
1356                start = String.valueOf(nums[k]);
1357
1358                if (potentialfurtherAscendingBeyondThisStart=="" && potentialfurtherAscendingBeyondThisEnd=="")
1359                {
1360                    sm.add(start);
1361                    System.out.println("019238475Writing Standalone: " + start);
1362                    completeTicker(start, start,k,lengthNums);
1363                }
1364
```

```
3.4f,3.4f,3.3f,3.3f,3.3f,3.4f
```

No ascendency, so it enters loop. We need to narrow scope and exclude if values are identical
But how can we be sure that 3.4f->3.3f has already been written as part of summary. There is an associated else statement for this

Implemented following code.

```
1353            if (!isPrevNumAscending) //careful
1354            {   //new code
1355                System.out.println("REACH HERE !!!!!!!!");
1356                start = String.valueOf(nums[k]);
1357
1358                if (potentialfurtherAscendingBeyondThisStart=="" && potentialfurtherAscendingBeyondThisEnd=="")
1359                {
1360                    if (k>0)
1361                    {
1362                    if (nums[k]!=nums[k-1])
1363                    {
1364                    sm.add(start);
1365                    System.out.println("019238475Writing Standalone: " + start);
1366                    completeTicker(start, start,k,lengthNums);
1367                    }
1368                    }
1369                }
```

```
[3.4->3.3, 3.3, 3.3->3.3, 3.3->3.4]
```

I now have to focus on this removal

```
CHECKING: 3.3 with 3.4


currently in list: [3.4->3.3, 3.3]
k=nums.length-2
3mWriting range: 3.3-> 3.3
asc counter: 0
desc counter: 1
ascending/descending next number (diffference)
49Writing range: 3.3-> 3.4
asc counter: 0
desc counter: 1
asc counter: 0
desc counter: 1

TICKER: SSA(2)
[3.4->3.3, 3.3, 3.3->3.3, 3.3->3.4]
Standalone numbers: 2   Ascending chains: 1   Descending chains: 0      TOTAL: 3


** Process exited - Return Code: 0 **
```

This is the unwanted part

OK

```
3.4f,3.4f,3.3f,3.3f,3.3f,3.4f
```

I am in the below section since:
Counter!=0
Store is empty.
Firstly I can observe this seems to be a distinct area of code.
I have created strong reference to the length of numbers and also looked two places back..
I am bit concerned if I change this without consideration.. So I will keep copy of this block and commented out

```
911                    //System.out.println("reach here");
912
913                    else
914                    {
915                        if (lengthNums>=4)
916                        {
917                            if (nums[k]==nums[k-2])
918                            {
919                                start=String.valueOf(nums[k-1]);
920                                end=String.valueOf(nums[k]);
921                                sm.add(start+"->"+end);
922                                System.out.println("3mWriting range: " + start + "-> " + end);
923                            }
924                        }
925                    }
926
```

```
929                    else
930                    {
931                        if (nums[k]!=nums[k-1])
932                        {
933                            start=String.valueOf(nums[k-1]);
934                            end=String.valueOf(nums[k]);
935                            sm.add(start+"->"+end);
936                            System.out.println("3mWriting range: " + start + "-> " + end);
937                        }
938                    }
```

```
[3.4->3.3, 3.3, 3.3->3.4]
```

```
4.7f,4.8f,4.9f,5.0f,5.0f,5.0f,4.9f,3.3f,3.2f,3.4f,3.3f,3.4f,3.2f
```

```
[4.7->5.0, 5.0->4.9, 3.3->3.2, 3.4->3.3, 3.4->3.3, 3.2]
```

```
CHECKING: 3.3 with 3.2


currently in list: [4.7->5.0, 5.0->4.9]
Descending sequence (difference)
3.3
3.2
-------------------------------------12Stored start -> end: 3.3->3.2
Establishing start: 3.3
19821982using stored start
197618Writing range: 3.3->3.2

CHECKING: 3.2 with 3.4


currently in list: [4.7->5.0, 5.0->4.9, 3.3->3.2]
K!=nums.length-2
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)

CHECKING: 3.4 with 3.3


currently in list: [4.7->5.0, 5.0->4.9, 3.3->3.2]
Descending sequence (difference)
3.4
3.3
-------------------------------------12Stored start -> end: 3.4->3.3
Establishing start: 3.4
19821982using stored start
197618Writing range: 3.4->3.3
```

This is ok

This is ok

This is ok storage

This is ok since it checks that nums[k+1] (3.3) == potential...End (3.3)

```
CHECKING: 3.3 with 3.4


currently in list: [4.7->5.0, 5.0->4.9, 3.3->3.2, 3.4->3.3]
K!=nums.length-2
1next number not descending (difference)
This is counter at the moment:  1
counter is not zero
Transition number: 3.3 ascending(difference) on either side
next number ascending (difference) AND previous number descending (difference
COUNTER VALUE: 1
previous number ascending (difference) AND next number ascending (difference)
-------------------------------------Stored start -> end: 3.4->3.3
```

This appears back to front, to be investigated..

```
CHECKING: 3.4 with 3.2
3.4
3.3
currently in list: [4.7->5.0, 5.0->4.9, 3.3->3.2, 3.4->3.3]
k=nums.length-2
previous number descending (difference)
6111using stored start
197618Writing range: 3.4->3.3
asc counter: 1
desc counter: 3
test: 3
11
13
false
test1
0241Writing Standalone: 3.2
This should be last number
asc counter: 1
desc counter: 0
asc counter: 1
desc counter: 0

TICKER: A(3)D(4)S
[4.7->5.0, 5.0->4.9, 3.3->3.2, 3.4->3.3, 3.4->3.3, 3.2]
```

I have performed this change. I consider it major change, so have

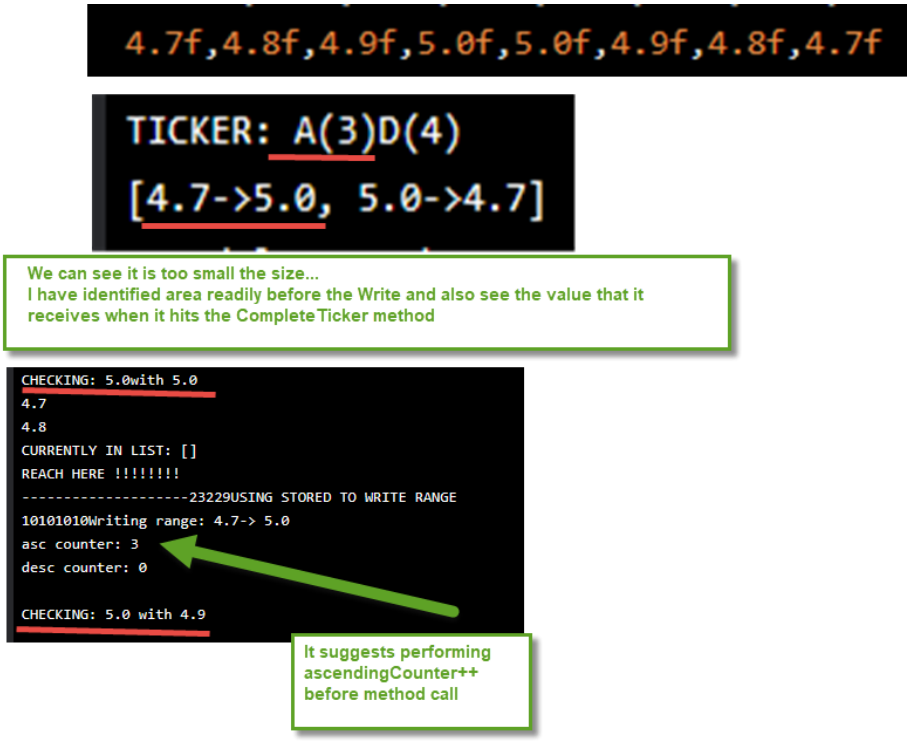
```
//potentialfurtherAscendingBeyondThisStart = start;   MAJOR CHANGE
//potentialfurtherAscendingBeyondThisEnd = end;  MAJOR CHANGE

potentialfurtherAscendingBeyondThisStart=String.valueOf(nums[k]);
potentialfurtherAscendingBeyondThisEnd=String.valueOf(nums[k+1]);
```

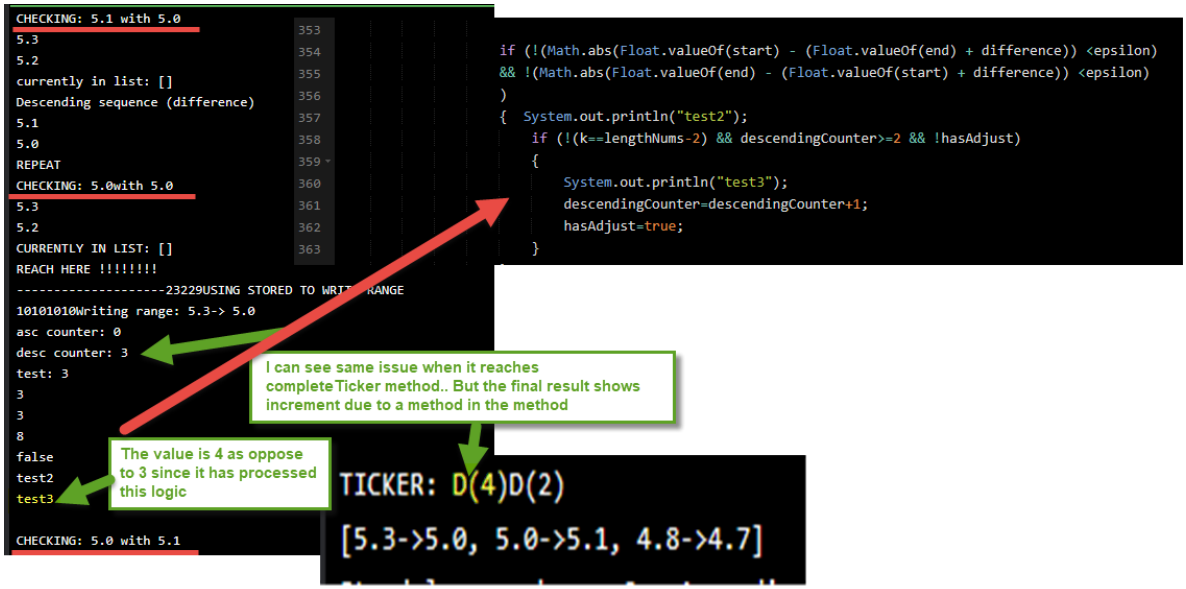
```
[4.7->5.0, 5.0->4.9, 3.3->3.2, 3.4->3.3, 3.3->3.4, 3.2]
```

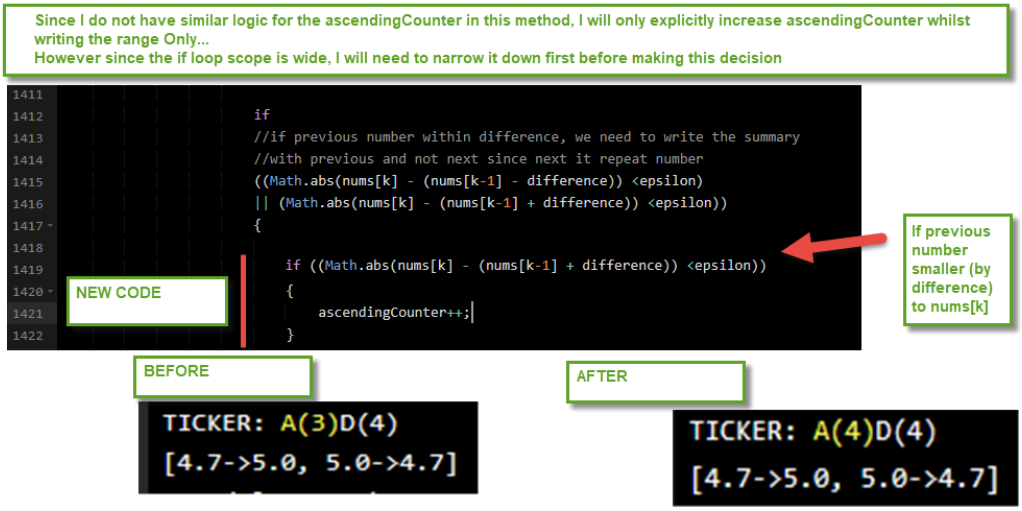
I now firmly believe that if all my test cases pass, I will be able to remediate my code to any test case which I have missed out.
It is quite impossible to prepare so many test cases and also ensure that I can test them manually correctly.

I have made several mass changes to the repeat number section...

However I am putting my focus onto the ticker for now.







I will now just go through all my test cases again devised in this document. I do not expect an adverse effect....

```
5.3f,5.2f,5.1f,5.0f,5.0f,5.1f,5.2f,5.3f
```

```
TICKER: D(4)A(3)
[5.3->5.0, 5.0->5.3]
```

This is too small

```
715
716
717
718
719
720
721
722
723
724
725
726
727
```

It can be seen that it has already increased counter, but it is still too short....

```
System.out.println("next number ascending (difference)");
System.out.println("------CURRENT LIST: " + sm);
end=String.valueOf(nums[k+1]);
sm.add(start+"->"+end);

System.out.println("35Writing range: " + start + "-> " + end);
ascendingCounter++;
completeTicker(start,end,k,lengthNums);
hasNotWrittenLastNumber=false;
isFirstOccurenceAscendingChain=false;
isFirstOccurenceAscendingChainNoTransition =false;
ascendingCounter++;
```

```
CHECKING: 5.2 with 5.3
5.1
5.2
currently in list: [5.3->5.0]
k=nums.length-2
next number ascending (difference)
------CURRENT LIST: [5.3->5.0]
35Writing range: 5.0-> 5.3
asc counter: 2
desc counter: 0
asc counter: 1
desc counter: 0
```

This suggests to me that I need to write something in the completeTicker method for this circumstance and perform an increment....

I will also need to investigate this, but its unrelated

```
CHECKING: 5.0 with 5.1
---------------------------VALS a: 0


currently in list: [5.3->5.0]
K!=nums.length-2
1next number not descending (difference)
This is counter at the moment:  3
counter is not zero
COUNTER VALUE: 3
previous number AND/OR next number not ascending(difference)

CHECKING: 5.1 with 5.2
---------------------------VALS a: 0


currently in list: [5.3->5.0]
K!=nums.length-2
1next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
2next number not descending (difference)
previous number descending(difference) AND/OR next number ascending (difference)
Next number ascending
First occurrence three consecutive ascending numbers (difference)
-------------------------------------22Stored start -> end: 5.1->5.2
Previous number descending (difference) AND next number ascending (difference)

CHECKING: 5.2 with 5.3
---------------------------VALS a: 1
5.1
5.2
currently in list: [5.3->5.0]
k=nums.length-2
next number ascending (difference)
------CURRENT LIST: [5.3->5.0]
35Writing range: 5.0-> 5.3
asc counter: 2
desc counter: 0
asc counter: 1
desc counter: 0
```

ascendingCounter has not been incremented since first instance

It has still not incremented since there is no system output proving that it is ascending

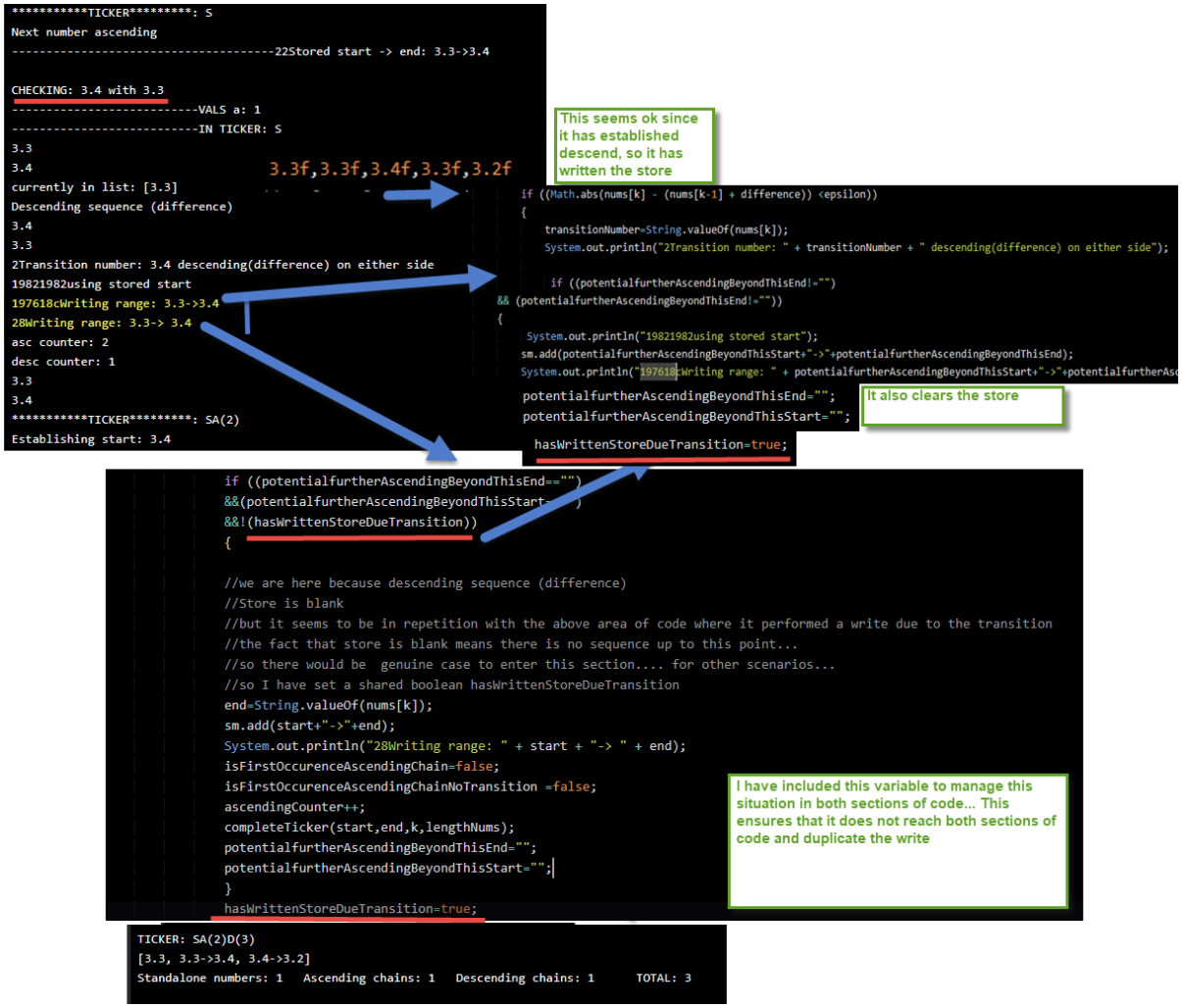
This is section of code in the ticker that causes counter to increase from 2=>3 However we still require another increment.. But first I need to understand this logic below here again

```
315
316
317
318
319
320
321
```

```
if ((k==lengthNums-2) && !hasAdjust)
{
    System.out.println("Test5");

    ascendingCounter=ascendingCounter+1;
}
```

```
35Writing range: 5.0-> 5.3
asc counter: 2
desc counter: 0
Test5
asc counter: 1
desc counter: 0

TICKER: D(4)A(3)
```

I have fixed above via tweaking my code and tested all new test cases..
I need to be very careful since there is one failed test case.. I need have full understanding of the scenario before making any change....



I had to use the same principle of providing mutual exclusiveness in similar concept to my test cases...

I had to introduce another variable since it was not related to having transition.

```
boolean hasWrittenRepeatNumber=false;
```

I had to remember that once it had been set to true, I had to prevent any areas of code entering by setting the boolean to true.

And once it passed the condition without entering, I was in a position to set the boolean hasWrittenRepeatNumber back to false, so that it is ready for the next number in the array with the initial state.

I am now going to try below. It is exact reversal of the sequence above..
I expect it to go through different area of code...

```
3.3f,3.3f,3.2f,3.3f,3.4f
```

It is meeting the standalone at front similar to the previous failed case

```
TICKER: D(2)A(3)
[3.3->3.2, 3.2->3.4]
Standalone numbers: 0   Ascending chains: 1   Descending chains: 1      TOTAL: 2
```

```
3.3f,3.3f,3.2f,3.3f,3.4f
```

```
REPEAT
CHECKING: 3.3with 3.3
--------------------------IN TICKER:


--------------------------VALS a: 0
CURRENTLY IN LIST: []
REACH HERE !!!!!!!!

CHECKING: 3.3 with 3.2
--------------------------VALS a: 0
--------------------------IN TICKER:


currently in list: []
Descending sequence (difference)
3.3
3.2
-------------------------------------12Stored start -> end: 3.3->3.2
Establishing start: 3.3

CHECKING: 3.2 with 3.3
--------------------------VALS a: 0
--------------------------IN TICKER:
3.3
3.2
currently in list: []
K!=nums.length-2
```

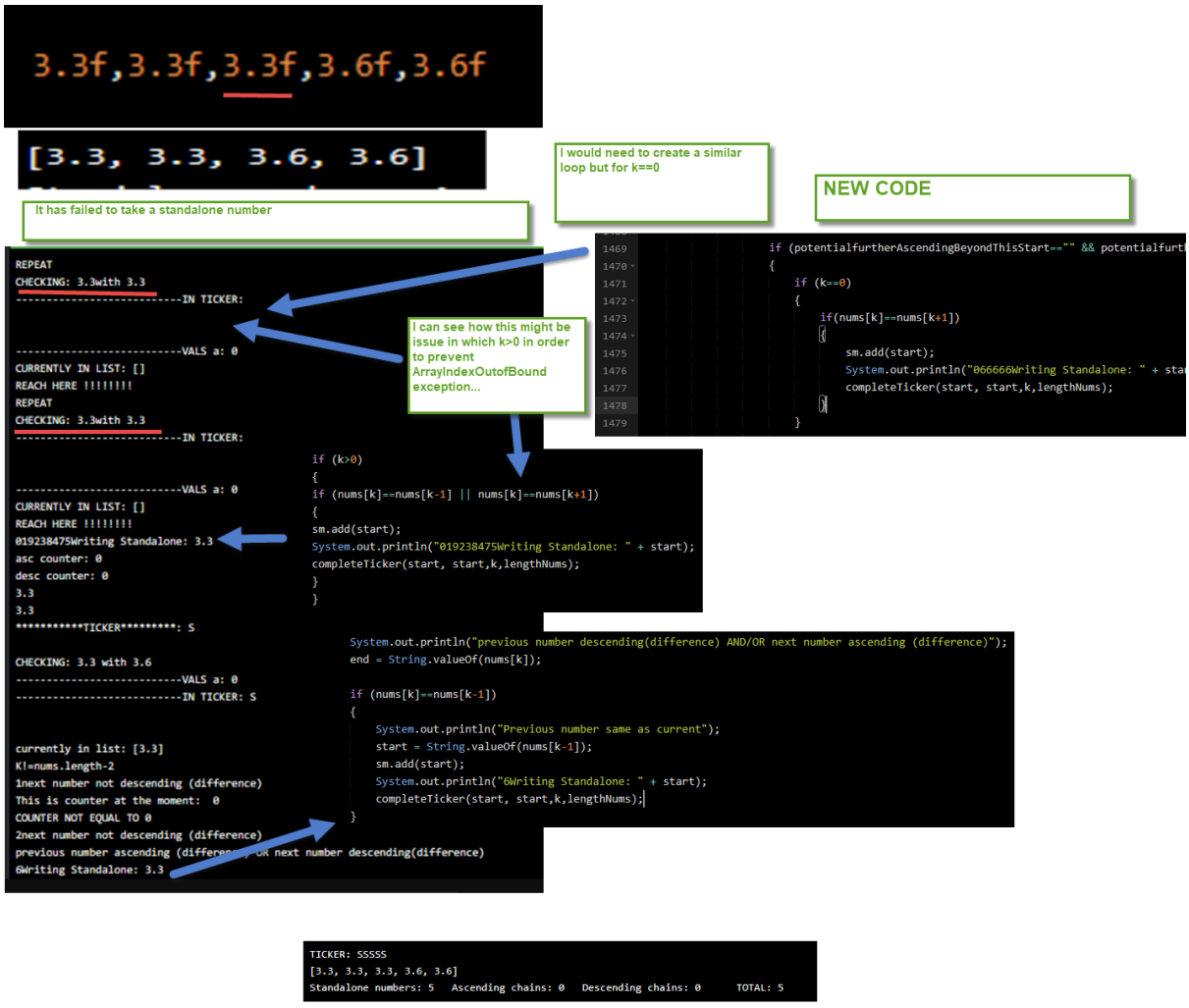
It is lacking logic to capture standalone. There are options to check if nums[k]=nums[k-1] However it is likely it might cause duplication elsewhere to writing into the List. We know other test cases are ok... If errors appear I will need to make the scope more

This is ok

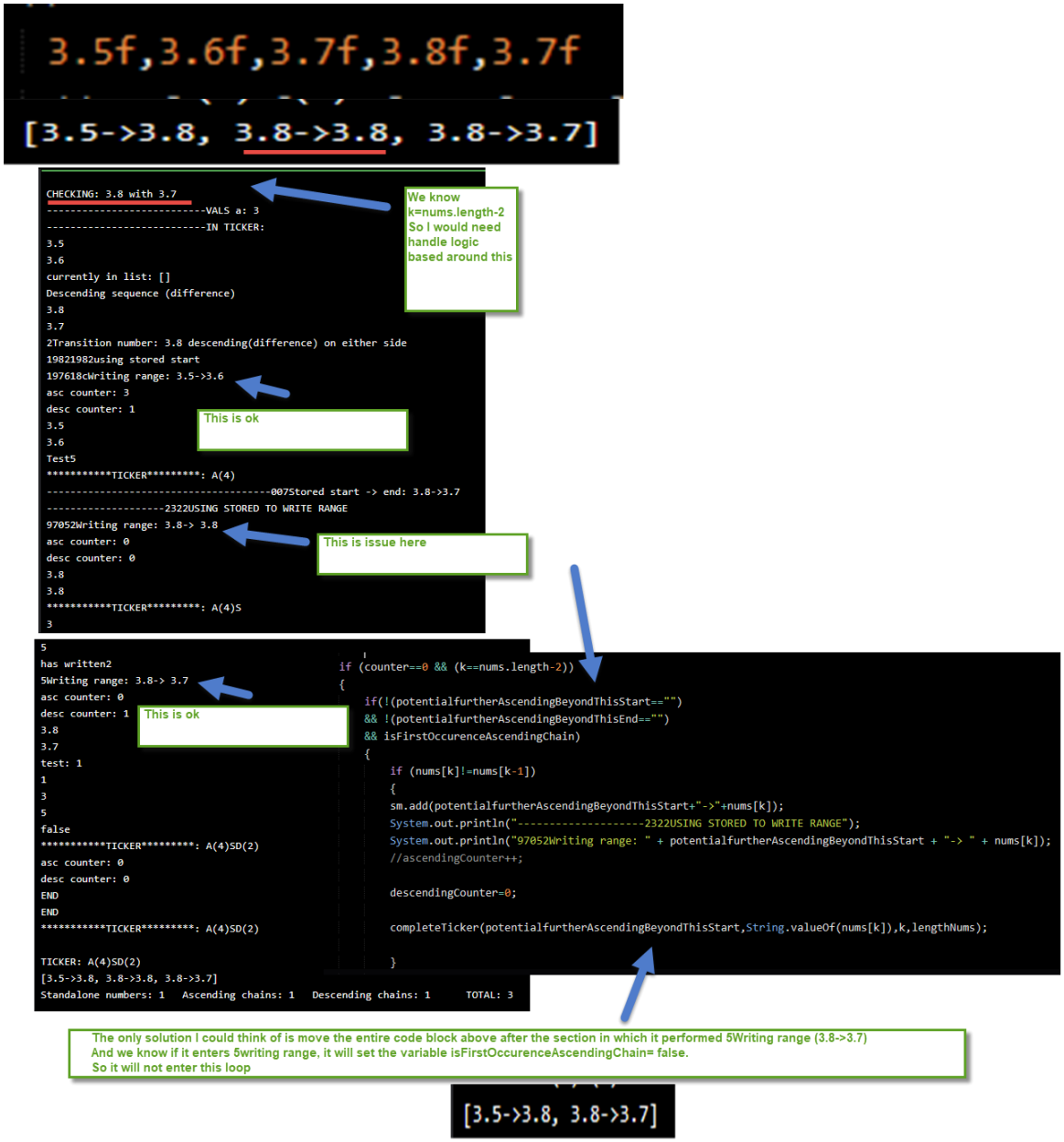
```
661
662
663                 if (nums[k]==nums[k-1])
664                 {
665                     System.out.println("0000Writing Standalone: " + start);
666                     sm.add(start);
667                     completeTicker(start, start,k,lengthNums);
668                 }
669
670
```

```
[3.3, 3.3->3.2, 3.2->3.4]
```

I have now extended the number standalones at front, it also required remediation...



I have now practically tested all my test cases in the documentation.
But there has been one failure...
It is extremely worrying to ensure nothing else is disrupted...
I need to also fully understand scenario so that I can also generate other scenarios resembling it.



I do not think it will have impact on my test cases.. In the worse case scenario, it is where k=nums.length -2

So I need to be careful and roll back the code should any issues occur

I found another issue also, again it is very specific and fortunately identifiable without having impact elsewhere.

I ran into several issues with ascending chain in between the ticker in which it was too short or too long.

I had to evaluate each of these scenarios:

```
//ISSUES THESE //Related to oversized or undersized ticker, it relates to transition and also standalone value occuring prior to it
   //35.1f, 35.2f, 35.3f, 35.2f, 35.1f, 85.6f  //ticker incorrect********************  //ok, ticker=ok  A D S
   //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=ok   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
//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
```

I had to re-work the ticker section and found this to be successful.
It of course creates a doubt for all my other test cases.. I know it will not change the List.
So I will need to check some previously checked test cases again...

There was dependency on previous outcome and also checking for transition into descendency going forward from nums[k]. I carried this logic forward as below

```
if ((tickerCounter==1 || tickerCounter==2)
&& (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon) && !hasAdjust)
{
    System.out.println("INCREASE1");
    ascendingCounter=ascendingCounter+1;
    System.out.println("NEW VAL: " + ascendingCounter);
    hasAdjust=true;
}

//3.5f,3.4f,3.35f,3.6f,3.7f,3.8f,3.7f,3.5f,3.6f,3.7f,3.8f,3.7f
System.out.println("----:"+previousOutcome);
System.out.println(nums[k]);
System.out.println(nums[k+1]);

 if ((previousOutcome=="standalone" || previousOutcome=="descending")
 && (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon)
 && !hasAdjust)
 {
     System.out.println("INSIDE HERE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
 ascendingCounter=ascendingCounter+1;
 hasAdjust=true;
 }

if ((k==lengthNums-2) && !hasAdjust)
{
```

I am also making a calculated late change in my code due to following test cases:

It failed to write the last range at end at position k=nums.length-2
I was quite nervous to perform this but I could find common area which would not trigger anything else... I also included a break at the end since it has written every single number in the array...

```
4.7f,4.8f,4.9f,5.0f,5.0f,4.9f  //ok, counter=ok
//4.7f,4.8f,4.9f,5.0f,4.8f,4.9f,5.0f,5.0f,4.9f  //ok, ticker=ok
```

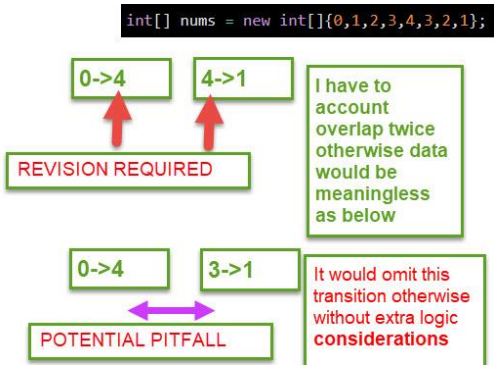
```
//we are here because we know k=nums.length-2 as per outer loop
//we know it hasWrittenRepeatNumber therefore is true....
//so it is best to narrow this situation down......
else
{
 if ((potentialfurtherAscendingBeyondThisStart=="")
&& (potentialfurtherAscendingBeyondThisEnd=="")
&& (hasWrittenRepeatNumber))

{
start=String.valueOf(nums[k]);
end=String.valueOf(nums[k+1]);
System.out.println("9711111Writing range: " + start + "-> " + end);
sm.add(start+"->" + end);
completeTicker(start,end,k,lengthNums);
break;
}
```

IMPORTANT

From all coding there is one are which can be confusing with respect to output

3.2f,3.3f,3.2f (3.2->3.3, 3.3->3.2)
3.2f,3.3f,3.3f,3.2f (3.2->3.3, 3.3->3.2)
We can see they are both summarised in exactly the same way…

Since when I started my coding I acknowledged the significance of providing the overlap to ensure no information is lost to end user..
And as can be seen on the above examples, we do not know which one has formed on the basis of



So only way is the information provided on overlap in the system output.
I have managed to modify my ticker, since when coming from descending sequence to ascending sequence…, if the next number after num[k] is ascending (difference), it is full consolidation that there is transition… I am using notation "-" to identify these on the ticker.
Likewise if the transition is ascending sequence, descending sequence, we expect next number after nums[k] to have descending difference.