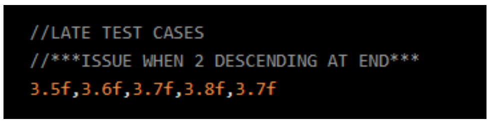
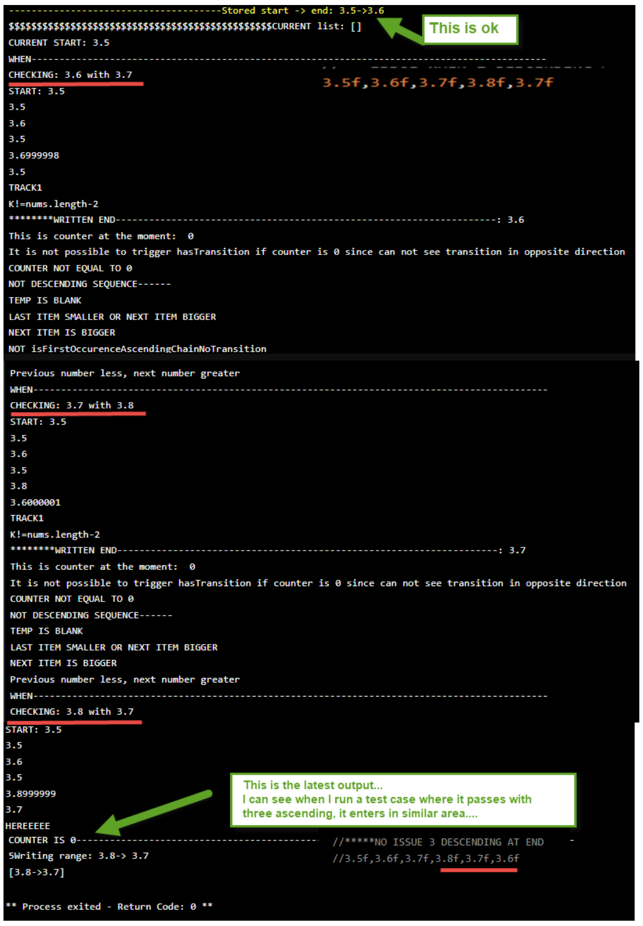
As always, late into testing. I thought of another test case as below.

TEST CASE:



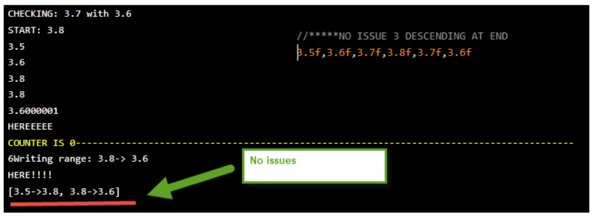
```
//LATE TEST CASES
//***ISSUE WHEN 2 DESCENDING AT END***
3.5f,3.6f,3.7f,3.8f,3.7f
```



```
--------------------------------------Stored start -> end: 3.5->3.6
$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$CURRENT list: []
CURRENT START: 3.5
WHEN-----------------------------------------------------------------------------------------------
CHECKING: 3.6 with 3.7                                    3.5f,3.6f,3.7f,3.8f,3.7f
START: 3.5
3.5
3.6
3.5
3.6999998
3.5
TRACK1
K!=nums.length-2
********WRITTEN END--------------------------------------------------------------------: 3.6
This is counter at the moment:  0
It is not possible to trigger hasTransition if counter is 0 since can not see transition in opposite direction
COUNTER NOT EQUAL TO 0
NOT DESCENDING SEQUENCE------
TEMP IS BLANK
LAST ITEM SMALLER OR NEXT ITEM BIGGER
NEXT ITEM IS BIGGER
NOT isFirstOccurenceAscendingChainNoTransition
Previous number less, next number greater
WHEN-----------------------------------------------------------------------------------------------
CHECKING: 3.7 with 3.8
START: 3.5
3.5
3.6
3.5
3.8
3.6000001
TRACK1
K!=nums.length-2
********WRITTEN END--------------------------------------------------------------------: 3.7
This is counter at the moment:  0
It is not possible to trigger hasTransition if counter is 0 since can not see transition in opposite direction
COUNTER NOT EQUAL TO 0
NOT DESCENDING SEQUENCE------
TEMP IS BLANK
LAST ITEM SMALLER OR NEXT ITEM BIGGER
NEXT ITEM IS BIGGER
Previous number less, next number greater
WHEN-----------------------------------------------------------------------------------------------
CHECKING: 3.8 with 3.7
START: 3.5
3.5
3.6
3.5
3.8999999
3.7
HEREEEEE
COUNTER IS 0-----------------------------------------      //*****NO ISSUE 3 DESCENDING AT END
5Writing range: 3.8-> 3.7                                  //3.5f,3.6f,3.7f,3.8f,3.7f,3.6f
[3.8->3.7]

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

This is ok

This is the latest output...
I can see when I run a test case where it passes with three ascending, it enters in similar area....

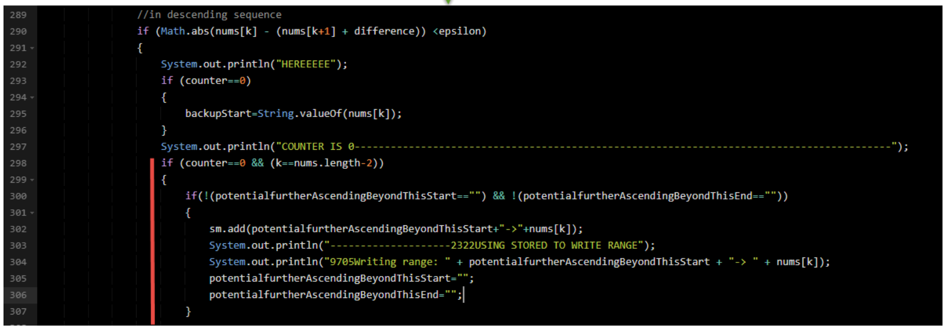


```
CHECKING: 3.7 with 3.6
START: 3.8                                    //*****NO ISSUE 3 DESCENDING AT END
3.5                                           3.5f,3.6f,3.7f,3.8f,3.7f,3.6f
3.6
3.8
3.8
3.6000001
HEREEEEE
COUNTER IS 0-----------------------------------------------------------------------------------
6Writing range: 3.8-> 3.6
HERE!!!!
[3.5->3.8, 3.8->3.6]
```

No issues

So I need to focus on the code in this area....
And most likely in the area of 5Writing range.
Perhaps I can check if the store variables are stilll populated
And if so, write the range.....
potentialfurtherAscendingBeyondThisStart -> nums[k]

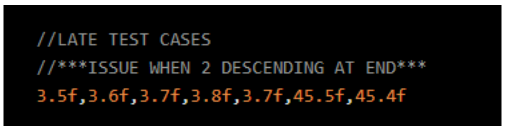
I have added this code



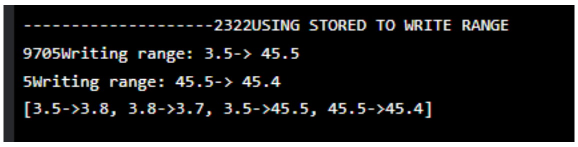
```java
//in descending sequence
if (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon)
{
    System.out.println("HEREEEEE");
    if (counter==0)
    {
        backupStart=String.valueOf(nums[k]);
    }
    System.out.println("COUNTER IS 0------------------------------------------------------------------------------------------");
    if (counter==0 && (k==nums.length-2))
    {
        if(!(potentialfurtherAscendingBeyondThisStart=="") && !(potentialfurtherAscendingBeyondThisEnd==""))
        {
            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
            System.out.println("-------------------2322USING STORED TO WRITE RANGE");
            System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
            potentialfurtherAscendingBeyondThisStart="";
            potentialfurtherAscendingBeyondThisEnd="";
        }
```

So now I have extended the test case to this and it fails..

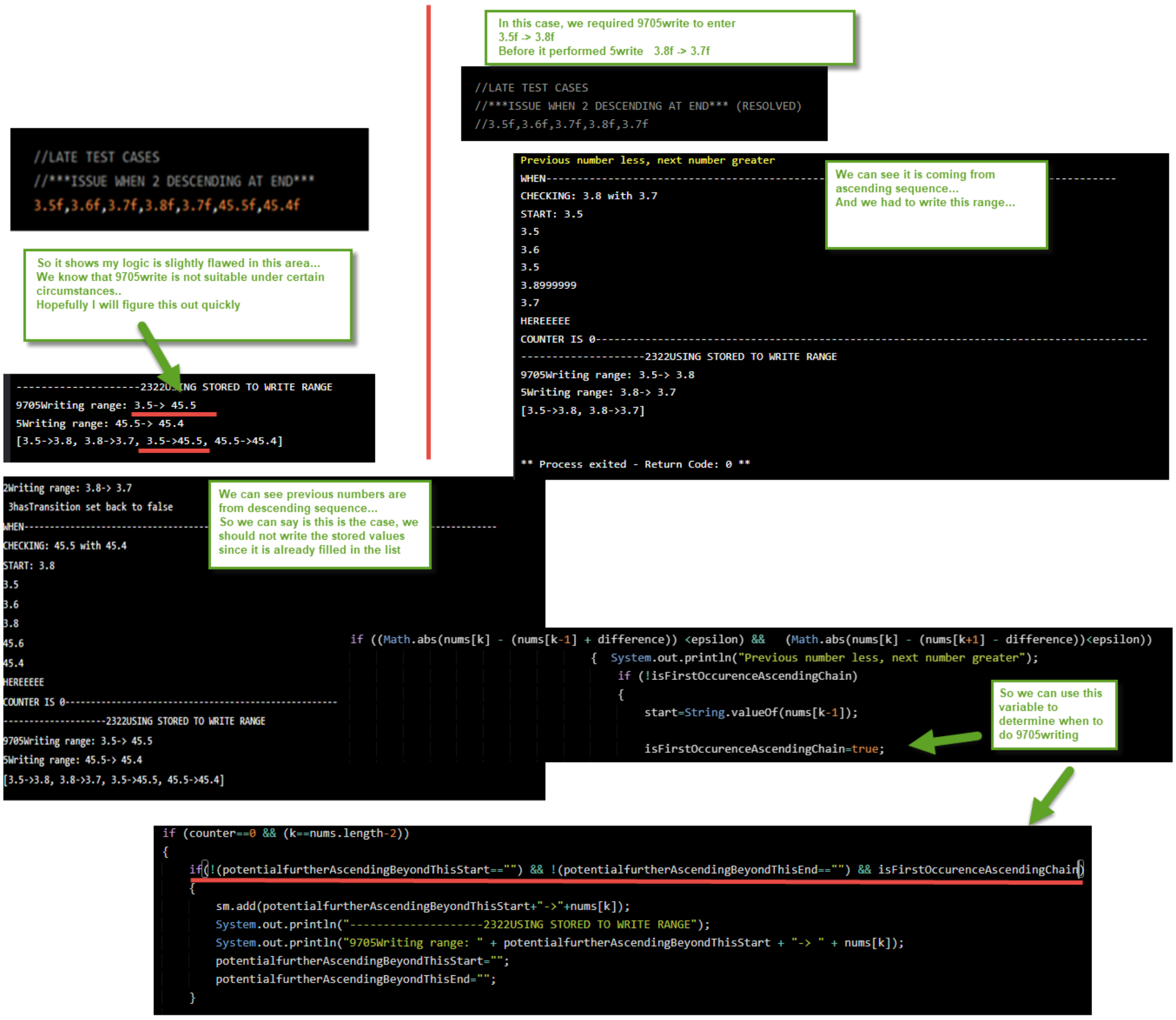
<u>TEST CASE:</u>



```
//LATE TEST CASES
//***ISSUE WHEN 2 DESCENDING AT END***
3.5f,3.6f,3.7f,3.8f,3.7f,45.5f,45.4f
```



```
--------------------2322USING STORED TO WRITE RANGE
9705Writing range: 3.5-> 45.5
5Writing range: 45.5-> 45.4
[3.5->3.8, 3.8->3.7, 3.5->45.5, 45.5->45.4]
```



In this case, we required 9705write to enter
3.5f -> 3.8f
Before it performed 5write   3.8f -> 3.7f

```
//LATE TEST CASES
//***ISSUE WHEN 2 DESCENDING AT END*** (RESOLVED)
//3.5f,3.6f,3.7f,3.8f,3.7f
```

```
Previous number less, next number greater
WHEN-----------------------------------------------------------------------------------
CHECKING: 3.8 with 3.7
START: 3.5
3.5
3.6
3.5
3.8999999
3.7
HEREEEEE
COUNTER IS 0------------------------------------------------------------------------------------------
--------------------2322USING STORED TO WRITE RANGE
9705Writing range: 3.5-> 3.8
5Writing range: 3.8-> 3.7
[3.5->3.8, 3.8->3.7]


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

We can see it is coming from ascending sequence...
And we had to write this range...

```
//LATE TEST CASES
//***ISSUE WHEN 2 DESCENDING AT END***
3.5f,3.6f,3.7f,3.8f,3.7f,45.5f,45.4f
```

So it shows my logic is slightly flawed in this area...
We know that 9705write is not suitable under certain circumstances..
Hopefully I will figure this out quickly

```
--------------------2322USING STORED TO WRITE RANGE
9705Writing range: 3.5-> 45.5
5Writing range: 45.5-> 45.4
[3.5->3.8, 3.8->3.7, 3.5->45.5, 45.5->45.4]
```

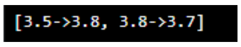
```
2Writing range: 3.8-> 3.7
 3hasTransition set back to false
WHEN-----------------------------------------------------------------------
CHECKING: 45.5 with 45.4
START: 3.8
3.5
3.6
3.8
45.6
45.4
HEREEEEE
COUNTER IS 0--------------------------------------------------
--------------------2322USING STORED TO WRITE RANGE
9705Writing range: 3.5-> 45.5
5Writing range: 45.5-> 45.4
[3.5->3.8, 3.8->3.7, 3.5->45.5, 45.5->45.4]
```

We can see previous numbers are from descending sequence...
So we can say is this is the case, we should not write the stored values since it is already filled in the list

```java
if ((Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon) &&   (Math.abs(nums[k] - (nums[k+1] - difference))<epsilon))
                                  {  System.out.println("Previous number less, next number greater");
                                     if (!isFirstOccurenceAscendingChain)
                                     {
                                        start=String.valueOf(nums[k-1]);

                                        isFirstOccurenceAscendingChain=true;
```

So we can use this variable to determine when to do 9705writing

```java
if (counter==0 && (k==nums.length-2))
{
    if(!(potentialfurtherAscendingBeyondThisStart=="") && !(potentialfurtherAscendingBeyondThisEnd=="") && isFirstOccurenceAscendingChain)
    {
        sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
        System.out.println("--------------------2322USING STORED TO WRITE RANGE");
        System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
        potentialfurtherAscendingBeyondThisStart="";
        potentialfurtherAscendingBeyondThisEnd="";
    }
```



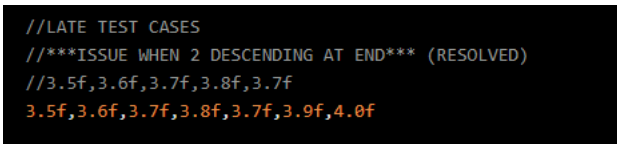
```
[3.5->3.8, 3.8->3.7]
```

<u>TEST CASE:</u>

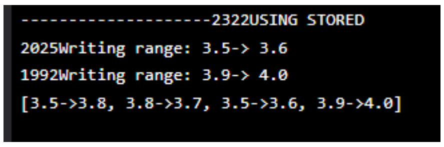
This is incorrect, it is writing 3.5->3.6

I have a feeling that in every place where it does a double range write...

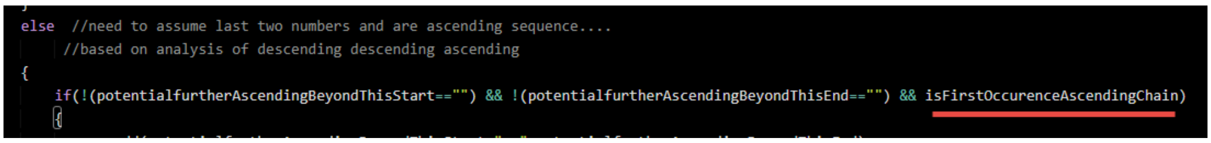
it has to look before it places the store value in for the state of



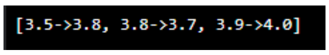
```
//LATE TEST CASES
//***ISSUE WHEN 2 DESCENDING AT END*** (RESOLVED)
//3.5f,3.6f,3.7f,3.8f,3.7f
3.5f,3.6f,3.7f,3.8f,3.7f,3.9f,4.0f
```



```
--------------------2322USING STORED
2025Writing range: 3.5-> 3.6
1992Writing range: 3.9-> 4.0
[3.5->3.8, 3.8->3.7, 3.5->3.6, 3.9->4.0]
```



```
else  //need to assume last two numbers and are ascending sequence....
      //based on analysis of descending descending ascending
{
    if(!(potentialfurtherAscendingBeyondThisStart=="") && !(potentialfurtherAscendingBeyondThisEnd=="") && isFirstOccurenceAscendingChain)
    {
```



```
[3.5->3.8, 3.8->3.7, 3.9->4.0]
```

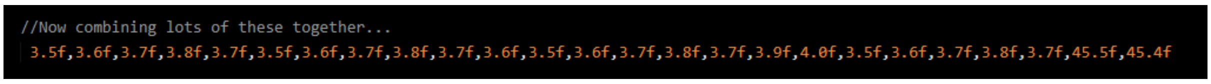
I will now go through all my new test cases and existing test cases..

I am just worried about when to clear the isFirstOccurenceAscendingChain since I can see in my code I have not set it to false on every time it uses the stored values...

I am not going to change my code blindly...

But I think I need to try lots ascending and descending small bursts in an array to determine if code is ok..

<u>TEST CASE:</u>



```
//Now combining lots of these together...
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,3.7f,3.9f,4.0f,3.5f,3.6f,3.7f,3.8f,3.7f,45.5f,45.4f
```

3.5->3.8*

3.8->3.7*

3.5->3.8*

3.8->3.5*

3.5->3.8*

3.8->3.7*

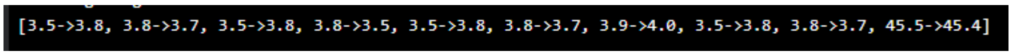
3.9->4.0 (this is correct, in my data it has 3.5->4.0

3.5->3.8 *

3.8->3.7*

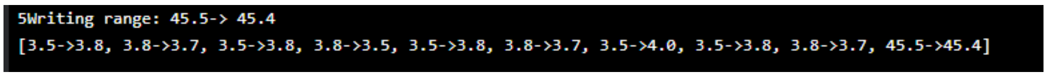
45.5->45.4*

This is in agreement with above....



```
[3.5->3.8, 3.8->3.7, 3.5->3.8, 3.8->3.5, 3.5->3.8, 3.8->3.7, 3.9->4.0, 3.5->3.8, 3.8->3.7, 45.5->45.4]
```

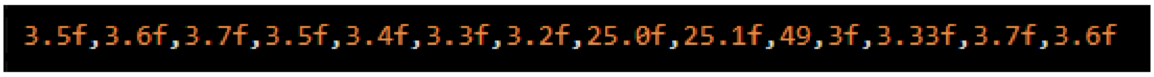
I also tried and set my code with resetting the stored values and Booleans at every point it made no difference. I do not want to implement changes without understanding for now...



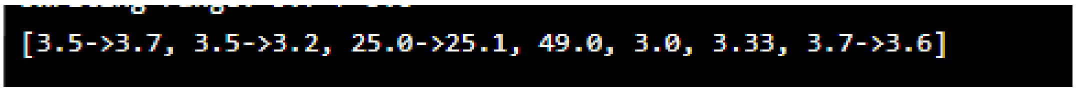
```
5Writing range: 45.5-> 45.4
[3.5->3.8, 3.8->3.7, 3.5->3.8, 3.8->3.5, 3.5->3.8, 3.8->3.7, 3.5->4.0, 3.5->3.8, 3.8->3.7, 45.5->45.4]
```

I will now try more test cases:

TEST CASE:



```
3.5f,3.6f,3.7f,3.5f,3.4f,3.3f,3.2f,25.0f,25.1f,49,3f,3.33f,3.7f,3.6f
```



```
[3.5->3.7, 3.5->3.2, 25.0->25.1, 49.0, 3.0, 3.33, 3.7->3.6]
```

This is fine....

I will go through my test cases again.. And unfortunately
any cases which start with ascending are still giving issues...

//***FAILS***

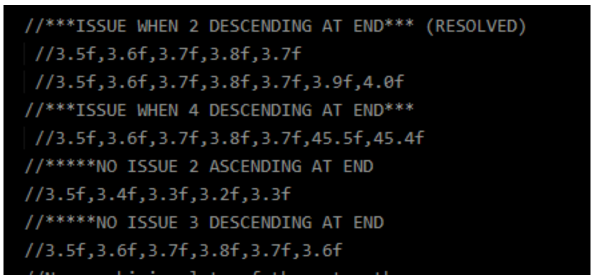
//3.5f,3.6f, 3.5f, 3.6f, 3.5f,3.4f,3.0f,2.9f,2.5f,2.4f //ascending descending ascending descending descending descending

//3.5f,3.6f,3.5f,3.1f,3.0f,2.9f //ascending descending descending

//3.5f,3.6f,3.5f,3.6f //ascending descending ascending

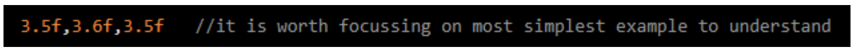
//3.5f,3.6f,3.5f,3.2f,3.1f //ascending descending descending

I know I fixed many cases similar from page 15 onwards which commence with ascending..... but I can see in these cases, there was a transitional descent...

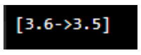


```
//***ISSUE WHEN 2 DESCENDING AT END*** (RESOLVED)
 //3.5f,3.6f,3.7f,3.8f,3.7f
 //3.5f,3.6f,3.7f,3.8f,3.7f,3.9f,4.0f
//***ISSUE WHEN 4 DESCENDING AT END***
 //3.5f,3.6f,3.7f,3.8f,3.7f,45.5f,45.4f
//*****NO ISSUE 2 ASCENDING AT END
//3.5f,3.4f,3.3f,3.2f,3.3f
//*****NO ISSUE 3 DESCENDING AT END
//3.5f,3.6f,3.7f,3.8f,3.7f,3.6f
```

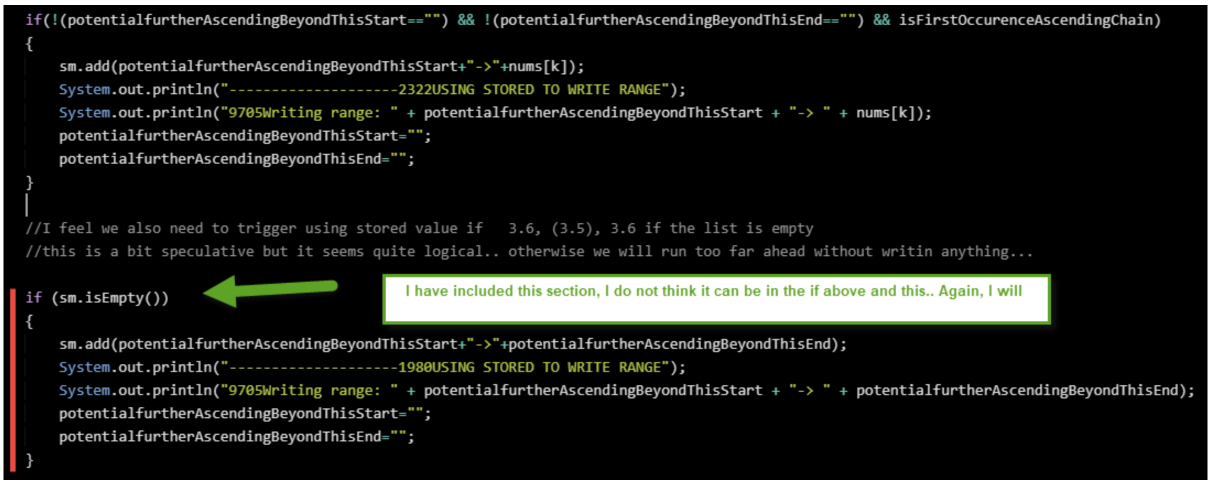
//I think the best option is to start with most basic failed case
<u>TEST CASE:</u>



```
3.5f,3.6f,3.5f   //it is worth focussing on most simplest example to understand
```

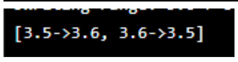


```
[3.6->3.5]
```



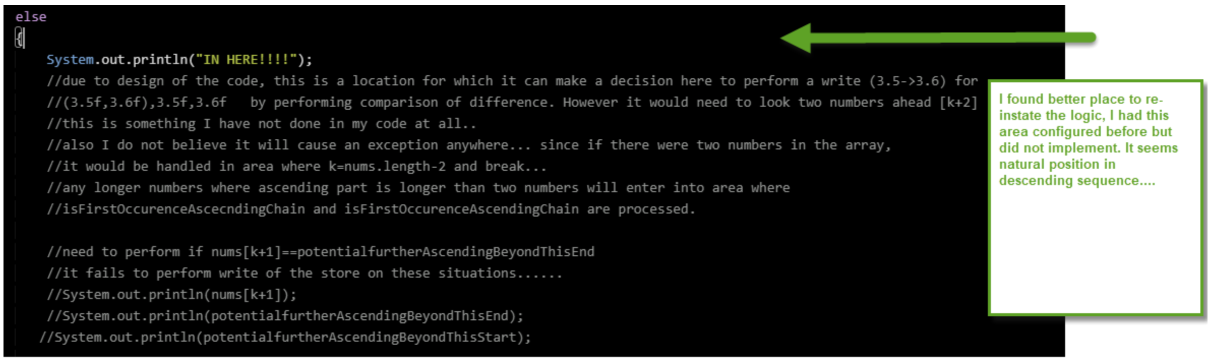
```
if(!(potentialfurtherAscendingBeyondThisStart=="") && !(potentialfurtherAscendingBeyondThisEnd=="") && isFirstOccurenceAscendingChain)
{
    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+nums[k]);
    System.out.println("-------------------2322USING STORED TO WRITE RANGE");
    System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
    potentialfurtherAscendingBeyondThisStart="";
    potentialfurtherAscendingBeyondThisEnd="";
}

//I feel we also need to trigger using stored value if   3.6, (3.5), 3.6 if the list is empty
//this is a bit speculative but it seems quite logical.. otherwise we will run too far ahead without writin anything...

if (sm.isEmpty())
{
    sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
    System.out.println("-------------------1980USING STORED TO WRITE RANGE");
    System.out.println("9705Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAscendingBeyondThisEnd);
    potentialfurtherAscendingBeyondThisStart="";
    potentialfurtherAscendingBeyondThisEnd="";
}
```

I have included this section, I do not think it can be in the if above and this.. Again, I will



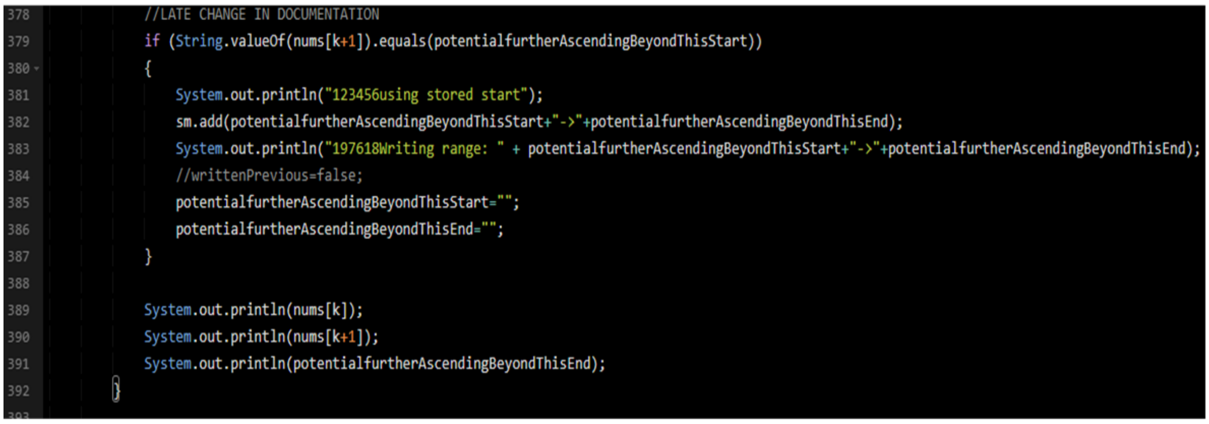
```
[3.5->3.6, 3.6->3.5]
```

I felt this was incorrect and too vague without reason.. So I followed code and implemented here:



```
else
{
    System.out.println("IN HERE!!!!");
    //due to design of the code, this is a location for which it can make a decision here to perform a write (3.5->3.6) for
    //(3.5f,3.6f),3.5f,3.6f   by performing comparison of difference. However it would need to look two numbers ahead [k+2]
    //this is something I have not done in my code at all..
    //also I do not believe it will cause an exception anywhere... since if there were two numbers in the array,
    //it would be handled in area where k=nums.length-2 and break...
    //any longer numbers where ascending part is longer than two numbers will enter into area where
    //isFirstOccurenceAscecndingChain and isFirstOccurenceAscendingChain are processed.

    //need to perform if nums[k+1]==potentialfurtherAscendingBeyondThisEnd
    //it fails to perform write of the store on these situations......
    //System.out.println(nums[k+1]);
    //System.out.println(potentialfurtherAscendingBeyondThisEnd);
   //System.out.println(potentialfurtherAscendingBeyondThisStart);
```

I found better place to re-instate the logic, I had this area configured before but did not implement. It seems natural position in descending sequence....



```
        //LATE CHANGE IN DOCUMENTATION
        if (String.valueOf(nums[k+1]).equals(potentialfurtherAscendingBeyondThisStart))
        {
            System.out.println("123456using stored start");
            sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
            System.out.println("197618Writing range: " + potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
            //writtenPrevious=false;
            potentialfurtherAscendingBeyondThisStart="";
            potentialfurtherAscendingBeyondThisEnd="";
        }

        System.out.println(nums[k]);
        System.out.println(nums[k+1]);
        System.out.println(potentialfurtherAscendingBeyondThisEnd);
    }
```

All my test cases pass.
I am hoping its finally resolved issues.
The last phase of course is testing it against the ChatGPT data.
And it has passed against all the ChatGPT extracts.....