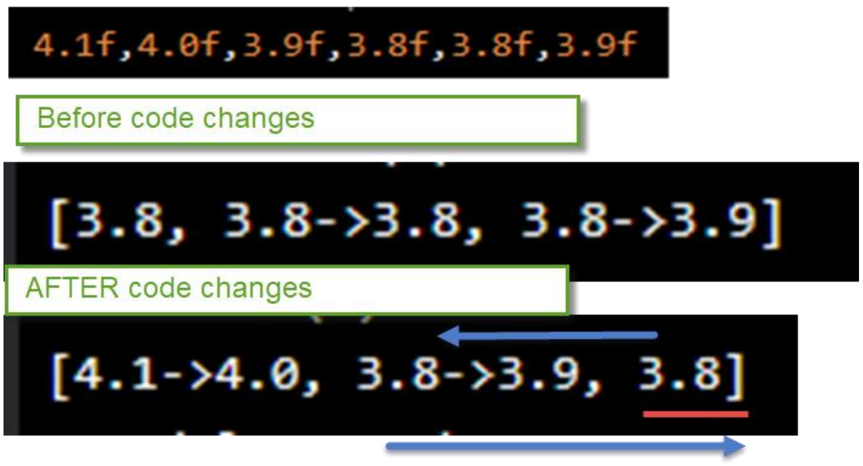
So I have performed same steps again as the previous document in order to resolve the output for:



```
4.1f,4.0f,3.9f,3.8f,3.8f,3.9f
```

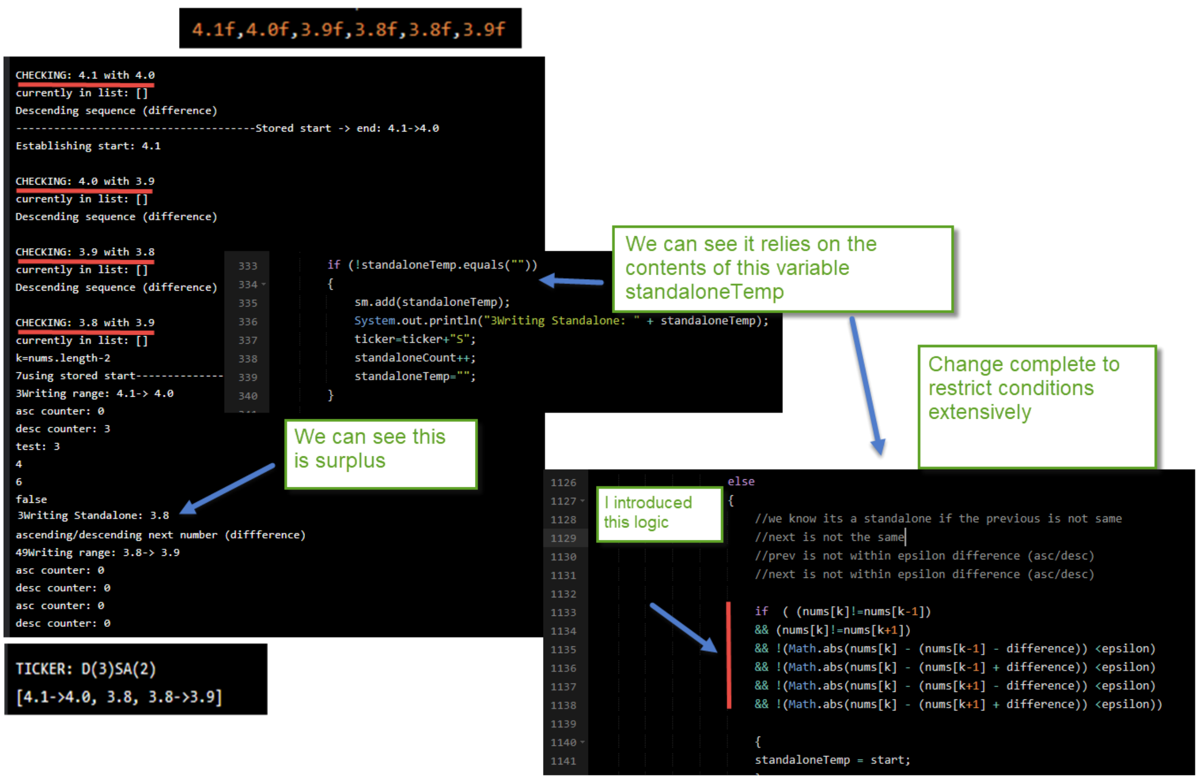
Before code changes

```
[3.8, 3.8->3.8, 3.8->3.9]
```

AFTER code changes

```
[4.1->4.0, 3.8->3.9, 3.8]
```

I fully understand the reason for these code changes… So it is a good time to try and figure out resolution from hereon.



```
4.1f,4.0f,3.9f,3.8f,3.8f,3.9f
```

```
CHECKING: 4.1 with 4.0
currently in list: []
Descending sequence (difference)
-------------------------------------Stored start -> end: 4.1->4.0
Establishing start: 4.1

CHECKING: 4.0 with 3.9
currently in list: []
Descending sequence (difference)

CHECKING: 3.9 with 3.8
currently in list: []
Descending sequence (difference)

CHECKING: 3.8 with 3.9
currently in list: []
k=nums.length-2
7using stored start-------------
3Writing range: 4.1-> 4.0
asc counter: 0
desc counter: 3
test: 3
4
6
false
 3Writing Standalone: 3.8
ascending/descending next number (diffference)
49Writing range: 3.8-> 3.9
asc counter: 0
desc counter: 0
asc counter: 0
desc counter: 0
```

```
TICKER: D(3)SA(2)
[4.1->4.0, 3.8, 3.8->3.9]
```

```java
    if (!standaloneTemp.equals(""))
    {
        sm.add(standaloneTemp);
        System.out.println("3Writing Standalone: " + standaloneTemp);
        ticker=ticker+"S";
        standaloneCount++;
        standaloneTemp="";
    }
```

We can see it relies on the contents of this variable standaloneTemp

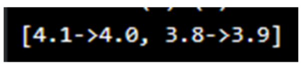
We can see this is surplus

Change complete to restrict conditions extensively

I introduced this logic

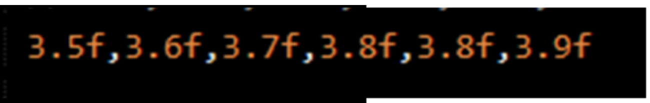
```java
else
{
    //we know its a standalone if the previous is not same
    //next is not the same
    //prev is not within epsilon difference (asc/desc)
    //next is not within epsilon difference (asc/desc)

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

    {
    standaloneTemp = start;
```

This already seems much more sensible than having to change the interconnection between all the methods to write standalone into the List

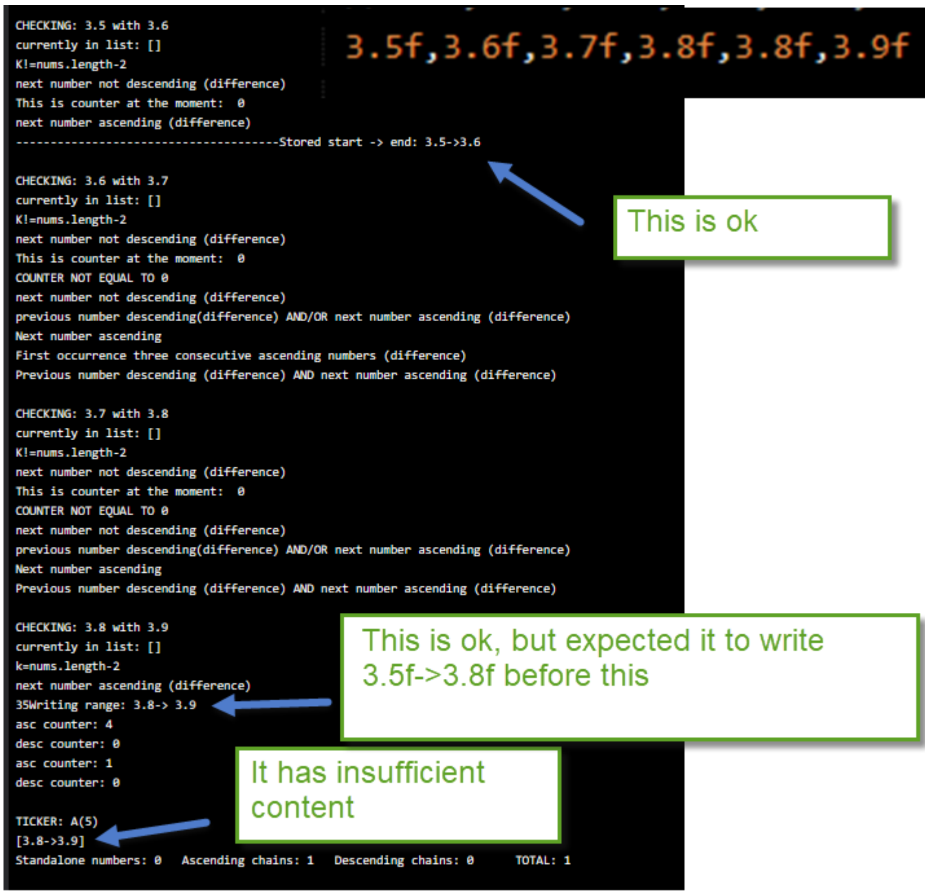


```
[4.1->4.0, 3.8->3.9]
```

I will not try the same again, but this time have leading ascending numbers.



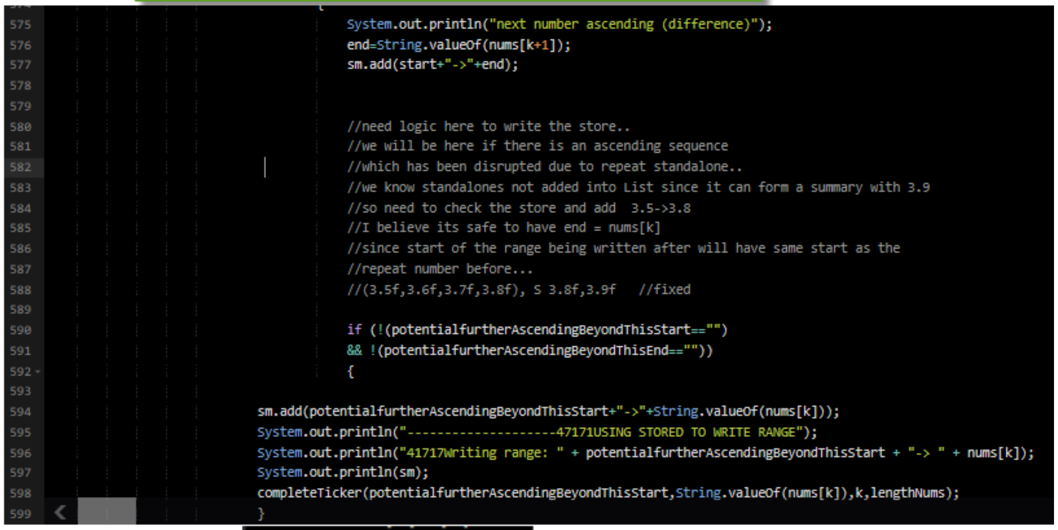
```
3.5f,3.6f,3.7f,3.8f,3.8f,3.9f
```



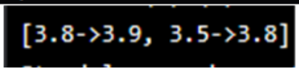
```
CHECKING: 3.5 with 3.6
currently in list: []
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
next number ascending (difference)
-----------------------------------Stored start -> end: 3.5->3.6

CHECKING: 3.6 with 3.7
currently in list: []
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next number not descending (difference)
previous number descending(difference) AND/OR next number ascending (difference)
Next number ascending
First occurrence three consecutive ascending numbers (difference)
Previous number descending (difference) AND next number ascending (difference)

CHECKING: 3.7 with 3.8
currently in list: []
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next 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: 3.8 with 3.9
currently in list: []
k=nums.length-2
next number ascending (difference)
35Writing range: 3.8-> 3.9
asc counter: 4
desc counter: 0
asc counter: 1
desc counter: 0

TICKER: A(5)
[3.8->3.9]
Standalone numbers: 0   Ascending chains: 1   Descending chains: 0      TOTAL: 1
```

This is ok
This is ok, but expected it to write 3.5f->3.8f before this
It has insufficient content

NEW CODE



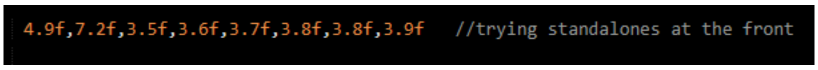
```java
                            System.out.println("next number ascending (difference)");
                            end=String.valueOf(nums[k+1]);
                            sm.add(start+"->"+end);


                            //need logic here to write the store..
                            //we will be here if there is an ascending sequence
                            //which has been disrupted due to repeat standalone..
                            //we know standalones not added into List since it can form a summary with 3.9
                            //so need to check the store and add  3.5->3.8
                            //I believe its safe to have end = nums[k]
                            //since start of the range being written after will have same start as the
                            //repeat number before...
                            //(3.5f,3.6f,3.7f,3.8f), S 3.8f,3.9f   //fixed

                            if (!(potentialfurtherAscendingBeyondThisStart=="")
                            && !(potentialfurtherAscendingBeyondThisEnd==""))
                            {

                sm.add(potentialfurtherAscendingBeyondThisStart+"->"+String.valueOf(nums[k]));
                System.out.println("-------------------47171USING STORED TO WRITE RANGE");
                System.out.println("41717Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
                System.out.println(sm);
                completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
                }
```



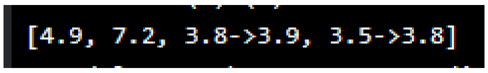
```
[3.8->3.9, 3.5->3.8]
```

My next phase is to now try a more complicated arrangement with standalone. Then I will try to introduce more intermittent standalone numbers...

TEST CASE:

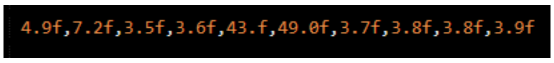


```
4.9f,7.2f,3.5f,3.6f,3.7f,3.8f,3.8f,3.9f   //trying standalones at the front
```

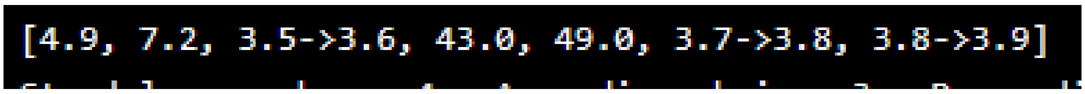


```
[4.9, 7.2, 3.8->3.9, 3.5->3.8]
```

TEST CASE:

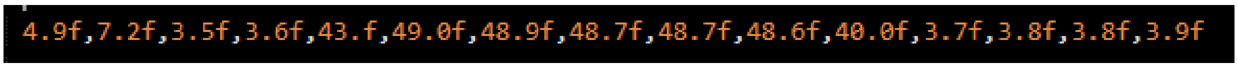


```
4.9f,7.2f,3.5f,3.6f,43.f,49.0f,3.7f,3.8f,3.8f,3.9f
```

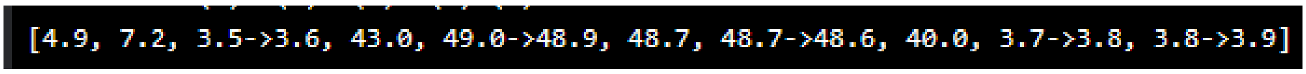


```
[4.9, 7.2, 3.5->3.6, 43.0, 49.0, 3.7->3.8, 3.8->3.9]
```

TEST CASE:



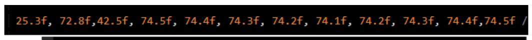
```
4.9f,7.2f,3.5f,3.6f,43.f,49.0f,48.9f,48.7f,48.7f,48.6f,40.0f,3.7f,3.8f,3.8f,3.9f
```



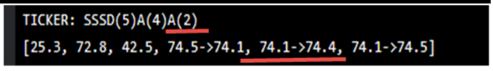
```
[4.9, 7.2, 3.5->3.6, 43.0, 49.0->48.9, 48.7, 48.7->48.6, 40.0, 3.7->3.8, 3.8->3.9]
```

It appears all is resolved..
I will try all test cases in software code again..

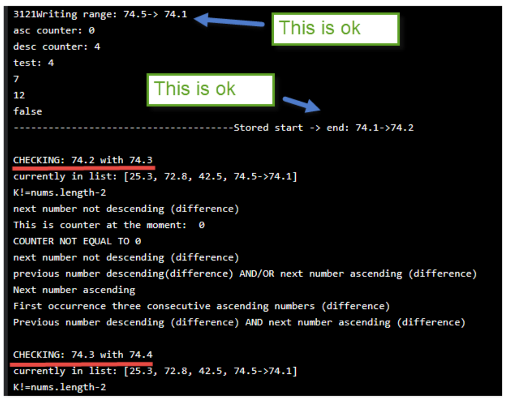
I have found a failing case as below:



```
25.3f, 72.8f,42.5f, 74.5f, 74.4f, 74.3f, 74.2f, 74.1f, 74.2f, 74.3f, 74.4f,74.5f /
```



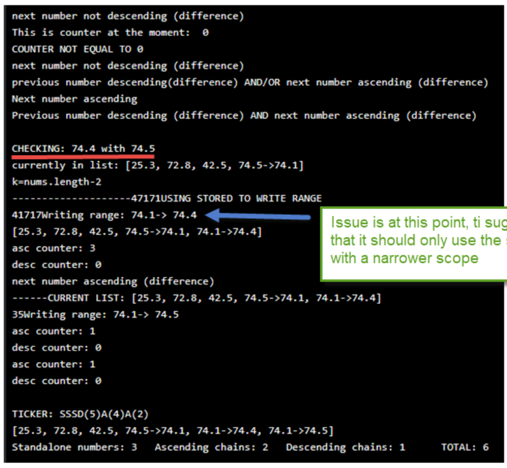
```
TICKER: SSSD(5)A(4)A(2)
[25.3, 72.8, 42.5, 74.5->74.1, 74.1->74.4, 74.1->74.5]
```



```
3121Writing range: 74.5-> 74.1
asc counter: 0
desc counter: 4
test: 4
7
12
false
------------------------------------Stored start -> end: 74.1->74.2

CHECKING: 74.2 with 74.3
currently in list: [25.3, 72.8, 42.5, 74.5->74.1]
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next number not descending (difference)
previous number descending(difference) AND/OR next number ascending (difference)
Next number ascending
First occurrence three consecutive ascending numbers (difference)
Previous number descending (difference) AND next number ascending (difference)

CHECKING: 74.3 with 74.4
currently in list: [25.3, 72.8, 42.5, 74.5->74.1]
K!=nums.length-2
```

This is ok

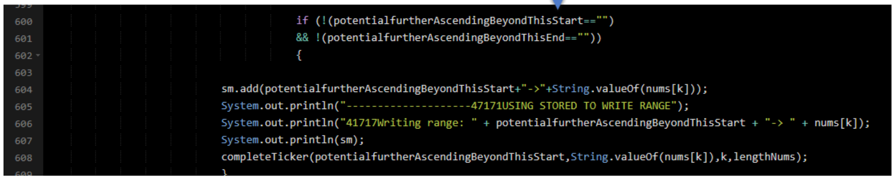
This is ok



```
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next 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: 74.4 with 74.5
currently in list: [25.3, 72.8, 42.5, 74.5->74.1]
k=nums.length-2
-------------------47171USING STORED TO WRITE RANGE
41717Writing range: 74.1-> 74.4
[25.3, 72.8, 42.5, 74.5->74.1, 74.1->74.4]
asc counter: 3
desc counter: 0
next number ascending (difference)
-----CURRENT LIST: [25.3, 72.8, 42.5, 74.5->74.1, 74.1->74.4]
35Writing range: 74.1-> 74.5
asc counter: 1
desc counter: 0
asc counter: 1
desc counter: 0

TICKER: SSSD(5)A(4)A(2)
[25.3, 72.8, 42.5, 74.5->74.1, 74.1->74.4, 74.1->74.5]
Standalone numbers: 3   Ascending chains: 2   Descending chains: 1      TOTAL: 6
```

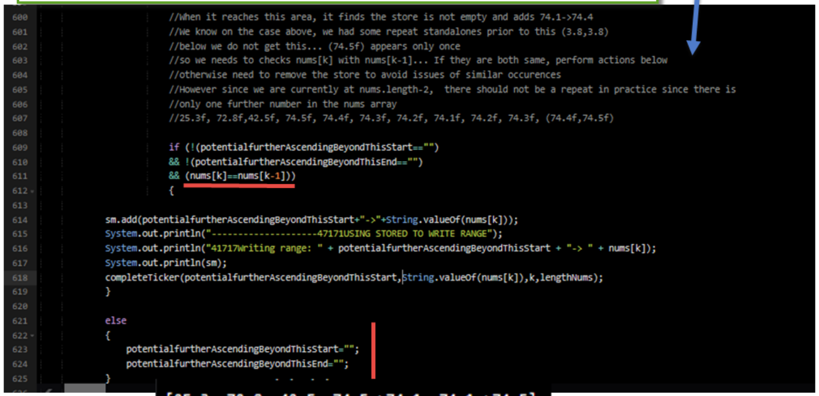
Issue is at this point, ti suggests that it should only use the store with a narrower scope

It just seems so likely that these variables have not been cleared. But I am not convinced this was the reason, this code was introduced to support sequence disrupted by standalones and another sequence appearing after.

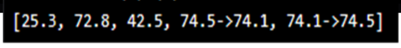


```java
                        if (!(potentialfurtherAscendingBeyondThisStart==""
                        && !(potentialfurtherAscendingBeyondThisEnd==""))
                        {

                sm.add(potentialfurtherAscendingBeyondThisStart+"->"+String.valueOf(nums[k]));
                System.out.println("-------------------47171USING STORED TO WRITE RANGE");
                System.out.println("41717Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
                System.out.println(sm);
                completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
                }
```

I have completed the change below



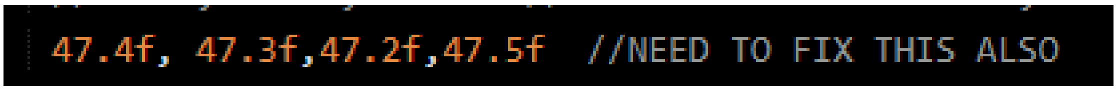
```java
                        //When it reaches this area, it finds the store is not empty and adds 74.1->74.4
                        //We know on the case above, we had some repeat standalones prior to this (3.8,3.8)
                        //below we do not get this... (74.5f) appears only once
                        //so we needs to checks nums[k] with nums[k-1]... If they are both same, perform actions below
                        //otherwise need to remove the store to avoid issues of similar occurences
                        //However since we are currently at nums.length-2,  there should not be a repeat in practice since there is
                        //only one further number in the nums array
                        //25.3f, 72.8f,42.5f, 74.5f, 74.4f, 74.3f, 74.2f, 74.1f, 74.2f, 74.3f, (74.4f,74.5f)

                        if (!(potentialfurtherAscendingBeyondThisStart=="")
                        && !(potentialfurtherAscendingBeyondThisEnd=="")
                        && (nums[k]==nums[k-1]))
                        {

                sm.add(potentialfurtherAscendingBeyondThisStart+"->"+String.valueOf(nums[k]));
                System.out.println("-------------------47171USING STORED TO WRITE RANGE");
                System.out.println("41717Writing range: " + potentialfurtherAscendingBeyondThisStart + "-> " + nums[k]);
                System.out.println(sm);
                completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
                }

                else
                {
                    potentialfurtherAscendingBeyondThisStart="";
                    potentialfurtherAscendingBeyondThisEnd="";
                }
```



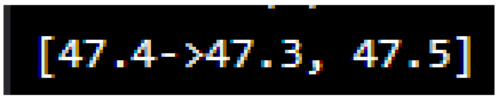
```
[25.3, 72.8, 42.5, 74.5->74.1, 74.1->74.5]
```

I almost am certain this was just a rare case since I had already reached k==nums.length-2
So I will go through all my test cases again....

We can see my change above has had adverse effect for scenario such as:

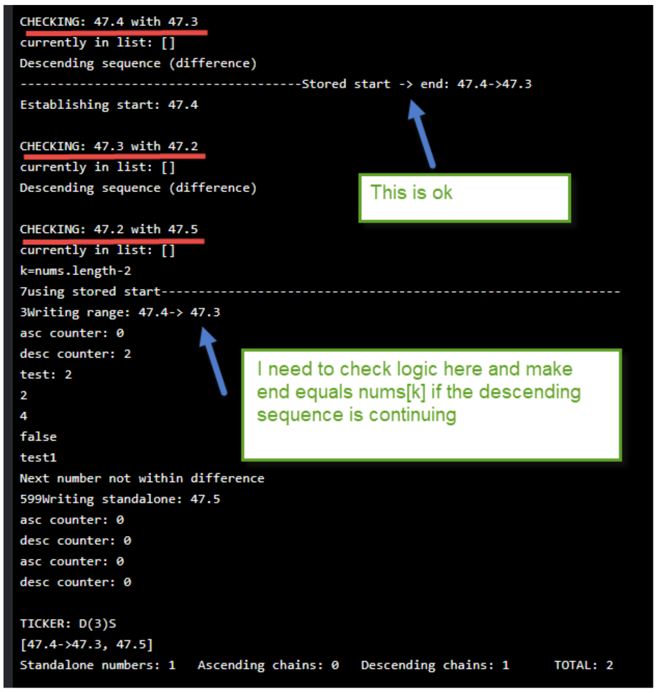


```
47.4f, 47.3f,47.2f,47.5f  //NEED TO FIX THIS ALSO
```



```
[47.4->47.3, 47.5]
```

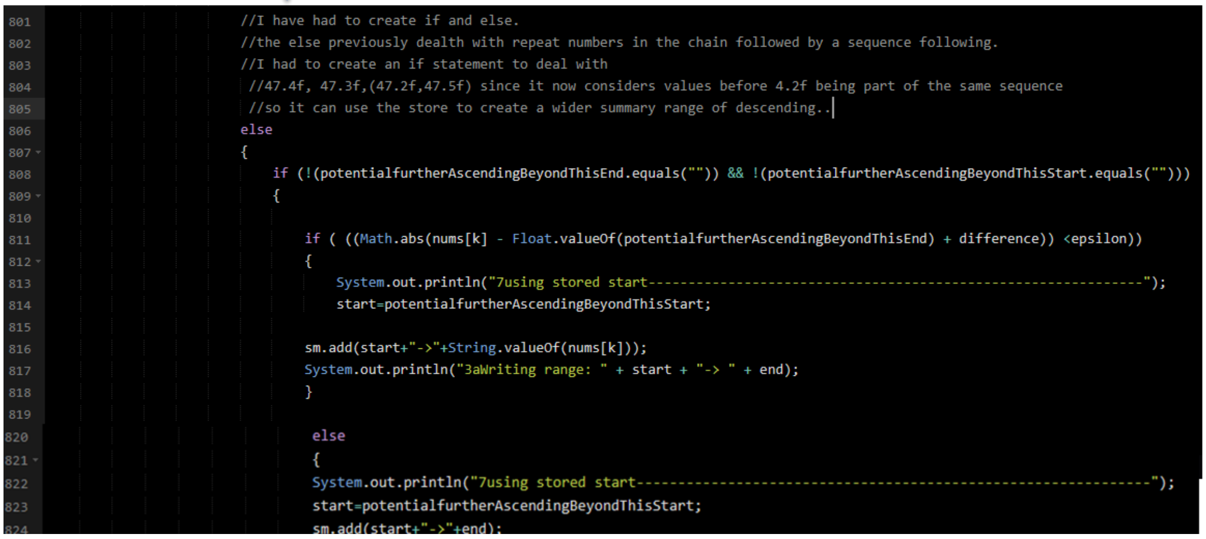
I will follow the code around again to understand this.



```
CHECKING: 47.4 with 47.3
currently in list: []
Descending sequence (difference)
------------------------------------Stored start -> end: 47.4->47.3
Establishing start: 47.4

CHECKING: 47.3 with 47.2
currently in list: []
Descending sequence (difference)

CHECKING: 47.2 with 47.5
currently in list: []
k=nums.length-2
7using stored start------------------------------------------------------------
3Writing range: 47.4-> 47.3
asc counter: 0
desc counter: 2
test: 2
2
4
false
test1
Next number not within difference
599Writing standalone: 47.5
asc counter: 0
desc counter: 0
asc counter: 0
desc counter: 0

TICKER: D(3)S
[47.4->47.3, 47.5]
Standalone numbers: 1   Ascending chains: 0   Descending chains: 1     TOTAL: 2
```

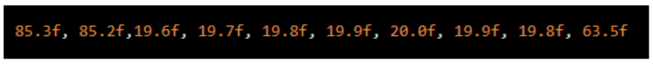
This is ok

I need to check logic here and make end equals nums[k] if the descending sequence is continuing

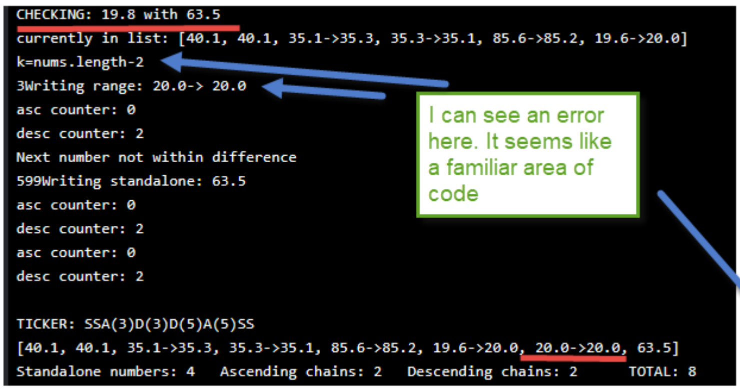


```
//I have had to create if and else.
//the else previously dealth with repeat numbers in the chain followed by a sequence following.
//I had to create an if statement to deal with
 //47.4f, 47.3f,(47.2f,47.5f) since it now considers values before 4.2f being part of the same sequence
 //so it can use the store to create a wider summary range of descending..
else
{
    if (!(potentialfurtherAscendingBeyondThisEnd.equals("")) && !(potentialfurtherAscendingBeyondThisStart.equals("")))
    {

        if ( ((Math.abs(nums[k] - Float.valueOf(potentialfurtherAscendingBeyondThisEnd) + difference)) <epsilon))
        {
            System.out.println("7using stored start------------------------------------------------------------");
            start=potentialfurtherAscendingBeyondThisStart;

        sm.add(start+"->"+String.valueOf(nums[k]));
        System.out.println("3aWriting range: " + start + "-> " + end);
        }

         else
         {
         System.out.println("7using stored start------------------------------------------------------------");
         start=potentialfurtherAscendingBeyondThisStart;
         sm.add(start+"->"+end);
```

I found an issue in area of code for k=nums.length-2



```
85.3f, 85.2f,19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f
```

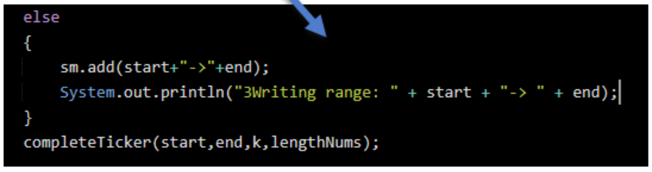


```
CHECKING: 19.8 with 63.5
currently in list: [40.1, 40.1, 35.1->35.3, 35.3->35.1, 85.6->85.2, 19.6->20.0]
k=nums.length-2
3Writing range: 20.0-> 20.0
asc counter: 0
desc counter: 2
Next number not within difference
599Writing standalone: 63.5
asc counter: 0
desc counter: 2
asc counter: 0
desc counter: 2

TICKER: SSA(3)D(3)D(5)A(5)SS
[40.1, 40.1, 35.1->35.3, 35.3->35.1, 85.6->85.2, 19.6->20.0, 20.0->20.0, 63.5]
Standalone numbers: 4    Ascending chains: 2    Descending chains: 2       TOTAL: 8
```

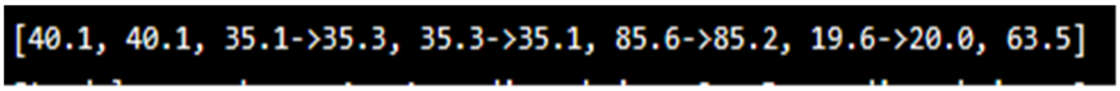
I can see an error here. It seems like a familiar area of code

It is here because there is no content in the store
So we need to be more clearer on why it is in this section of code

It might be surplus to requirements altogether



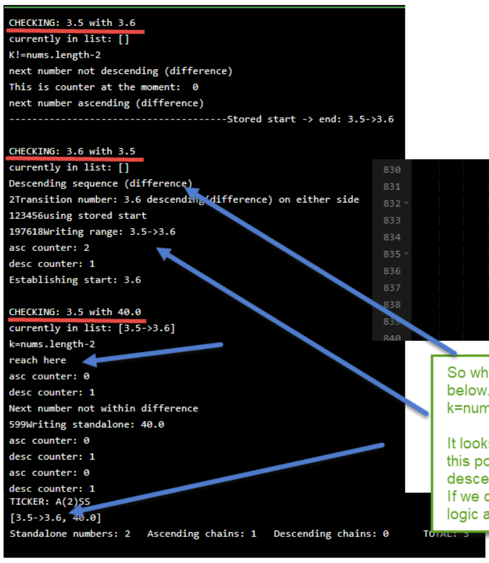
```
else
{
    sm.add(start+"->"+end);
    System.out.println("3Writing range: " + start + "-> " + end);
}
completeTicker(start,end,k,lengthNums);
```



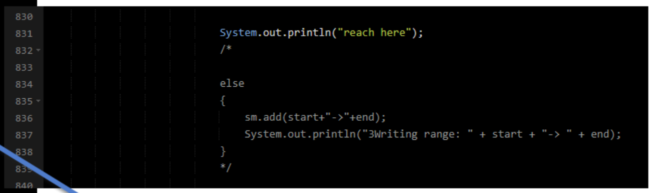
```
[40.1, 40.1, 35.1->35.3, 35.3->35.1, 85.6->85.2, 19.6->20.0, 63.5]
```

I will go through all my tests again, since its related to the ending part, I will not test the ChatGPT extracts again for now

Test case with issues....



```
CHECKING: 3.5 with 3.6
currently in list: []
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
next number ascending (difference)
------------------------------------Stored start -> end: 3.5->3.6

CHECKING: 3.6 with 3.5
currently in list: []
Descending sequence (difference)
2Transition number: 3.6 descending(difference) on either side
123456using stored start
197618Writing range: 3.5->3.6
asc counter: 2
desc counter: 1
Establishing start: 3.6

CHECKING: 3.5 with 40.0
currently in list: [3.5->3.6]
k=nums.length-2
reach here
asc counter: 0
desc counter: 1
Next number not within difference
599Writing standalone: 40.0
asc counter: 0
desc counter: 1
asc counter: 0
desc counter: 1
TICKER: A(2)SS
[3.5->3.6, 40.0]
Standalone numbers: 2   Ascending chains: 1   Descending chains: 0      TOTAL: 3
```



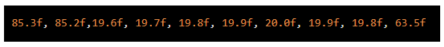
```
                        System.out.println("reach here");
                        /*

                        else
                        {
                            sm.add(start+"->"+end);
                            System.out.println("3Writing range: " + start + "-> " + end);
                        }
                        */
```

So why is this loop above required in this scenario and not in below. In both circumstances,
k=nums.length-2 and store is empty

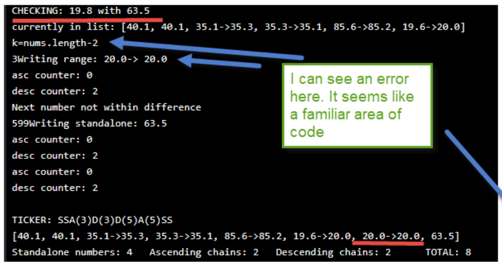
It looks like in practice it should have created the store again at this point.... since it has acknowledged also that it is a descending sequence...
If we decide against this, we can perhaps look to introduce more logic and re-instate the code below

In my code so far, I **never used the store and also written into it straight after**.. So it will be difficult to implement it into right section. So perhaps I can just use logic at bottom and re-instate the loop. But issue is I would need to look further back into the array. This will be the first time I will be looking greater than k+1 or k-1.
This seems like safest option unfortunately....

If nums[k] == nums[k-2]
i.e in a situation where it performs ascending descending
3.5, 3.6, 3.5 is applicable
Whereas
as 20.0f, 19.9f, 19.8f is not applicable



```
85.3f, 85.2f,19.6f, 19.7f, 19.8f, 19.9f, 20.0f, 19.9f, 19.8f, 63.5f
```

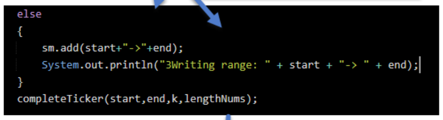


```
CHECKING: 19.8 with 63.5
currently in list: [40.1, 40.1, 35.1->35.3, 35.3->35.1, 85.6->85.2, 19.6->20.0]
k=nums.length-2
3Writing range: 20.0-> 20.0
asc counter: 0
desc counter: 2
Next number not within difference
599Writing standalone: 63.5
asc counter: 0
desc counter: 2
asc counter: 0
desc counter: 2

TICKER: SSA(3)D(3)D(5)A(5)SS
[40.1, 40.1, 35.1->35.3, 35.3->35.1, 85.6->85.2, 19.6->20.0, 20.0->20.0, 63.5]
Standalone numbers: 4   Ascending chains: 2   Descending chains: 2      TOTAL: 8
```

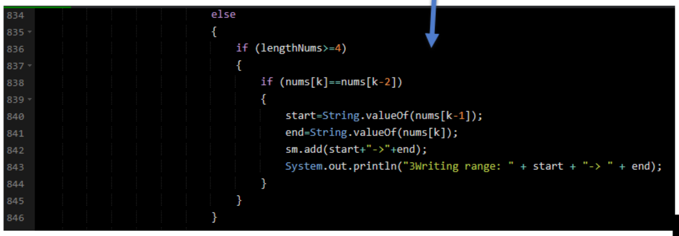
I can see an error here. It seems like a familiar area of code

It is here because there is no content in the store
So we need to be more clearer on why it is in this section of code

It might be surplus to requirements altogether



```
else
{
    sm.add(start+"->"+end);
    System.out.println("3Writing range: " + start + "-> " + end);
}
completeTicker(start,end,k,lengthNums);
```

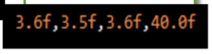


```
else
{
    if (lengthNums>=4)
    {
        if (nums[k]==nums[k-2])
        {
            start=String.valueOf(nums[k-1]);
            end=String.valueOf(nums[k]);
            sm.add(start+"->"+end);
            System.out.println("3Writing range: " + start + "-> " + end);
        }
    }
}
```

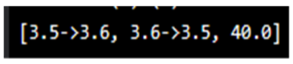
I have created because of the symmetry with nums[k-2], nums[k-1] and nums[k]

I am just not sure if 3.5, 3.6, 3.5 and also 3.6,3.5,3.6 are applicable

Fortunately does not enter same loop

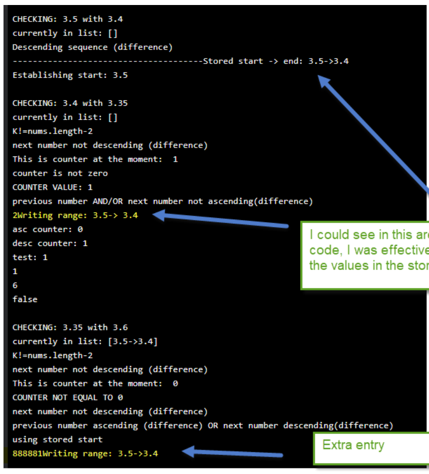


```
3.6f,3.5f,3.6f,40.0f
```



```
[3.5->3.6, 3.6->3.5, 40.0]
```

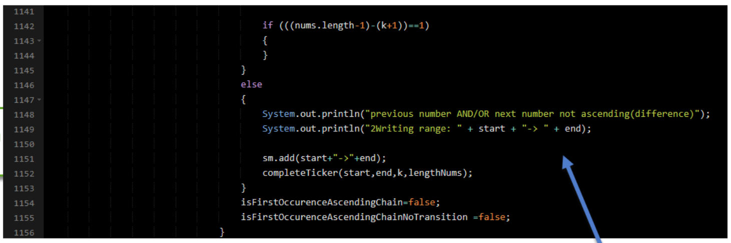
I had another failed case:



```
CHECKING: 3.5 with 3.4
currently in list: []
Descending sequence (difference)
-----------------------------------Stored start -> end: 3.5->3.4
Establishing start: 3.5

CHECKING: 3.4 with 3.35
currently in list: []
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)
2Writing range: 3.5-> 3.4
asc counter: 0
desc counter: 1
test: 1
1
6
false

CHECKING: 3.35 with 3.6
currently in list: [3.5->3.4]
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next number not descending (difference)
previous number ascending (difference) OR next number descending(difference)
using stored start
888881Writing range: 3.5->3.4
```

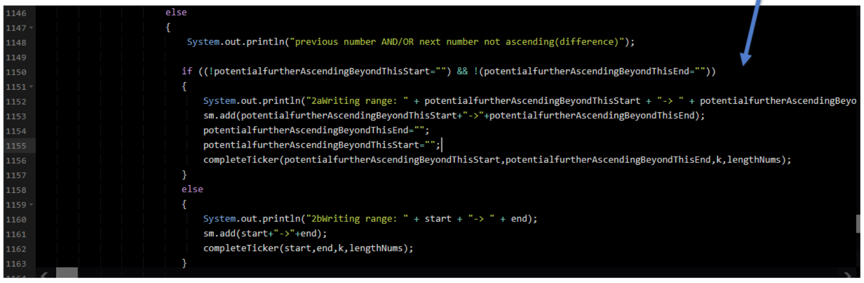
I could see in this area of code, I was effectively writing the values in the store

Extra entry

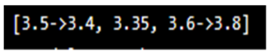


```
                    if (((nums.length-1)-(k+1))==1)
                    {
                    }
                }
                else
                {
                    System.out.println("previous number AND/OR next number not ascending(difference)");
                    System.out.println("2Writing range: " + start + "-> " + end);

                    sm.add(start+"->"+end);
                    completeTicker(start,end,k,lengthNums);
                }
                isFirstOccurenceAscendingChain=false;
                isFirstOccurenceAscendingChainNoTransition =false;
            }
```

But I had no formal process to use the store.. So I implemented code as below...



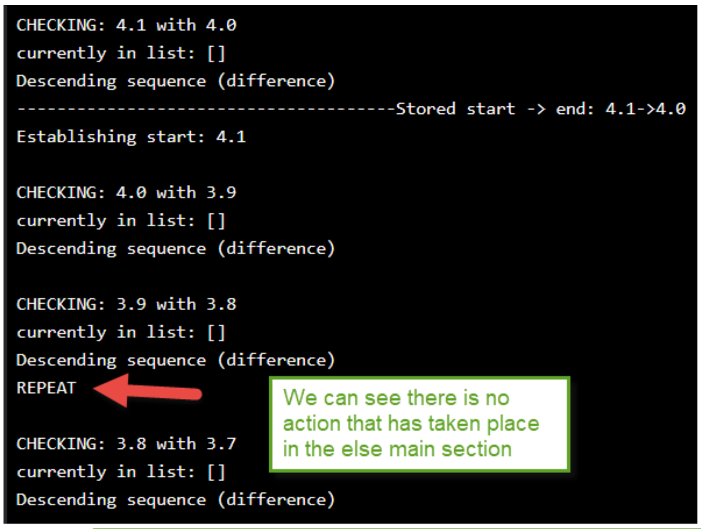
```
                else
                {
                 System.out.println("previous number AND/OR next number not ascending(difference)");

                    if ((!potentialfurtherAscendingBeyondThisStart="") && !(potentialfurtherAscendingBeyondThisEnd=""))
                    {
                        System.out.println("2aWriting range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurtherAscendingBeyo
                        sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
                        potentialfurtherAscendingBeyondThisEnd="";
                        potentialfurtherAscendingBeyondThisStart="";
                        completeTicker(potentialfurtherAscendingBeyondThisStart,potentialfurtherAscendingBeyondThisEnd,k,lengthNums);
                    }
                    else
                    {
                        System.out.println("2bWriting range: " + start + "-> " + end);
                        sm.add(start+"->"+end);
                        completeTicker(start,end,k,lengthNums);
                    }
```



```
[3.5->3.4, 3.35, 3.6->3.8]
```

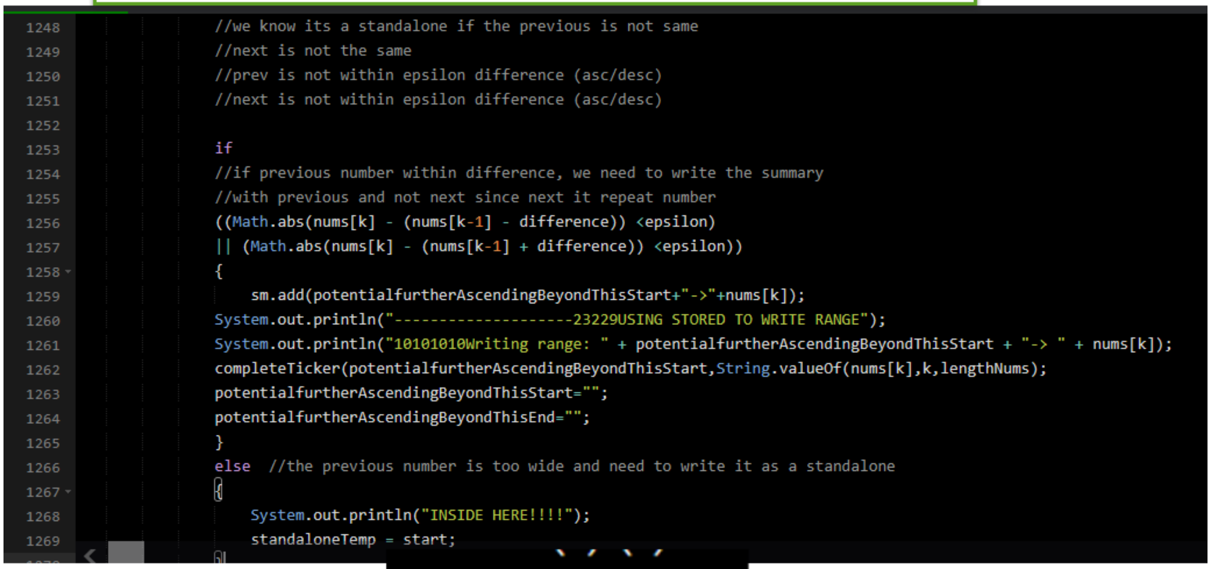
I will go through all my test cases again since I consider this critical change.

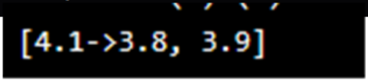
I have found a failing test case and straight away it leads me to believe that I did coding about to deal with standalones when k=nums.length-2


```
CHECKING: 4.1 with 4.0
currently in list: []
Descending sequence (difference)
-------------------------------------Stored start -> end: 4.1->4.0
Establishing start: 4.1

CHECKING: 4.0 with 3.9
currently in list: []
Descending sequence (difference)

CHECKING: 3.9 with 3.8
currently in list: []
Descending sequence (difference)
REPEAT

CHECKING: 3.8 with 3.7
currently in list: []
Descending sequence (difference)
```

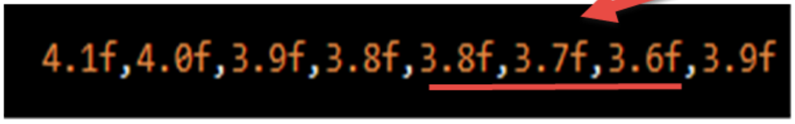
We can see there is no action that has taken place in the else main section

I analysed my code again in this section, and it has given a closer result..

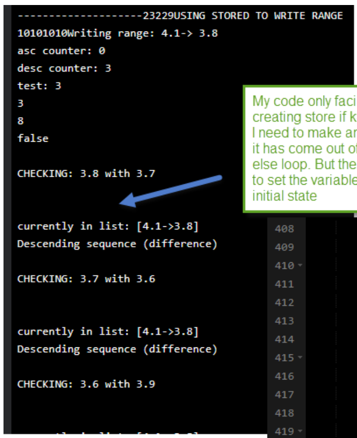

```java
                //we know its a standalone if the previous is not same
                //next is not the same
                //prev is not within epsilon difference (asc/desc)
                //next is not within epsilon difference (asc/desc)

                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);
                potentialfurtherAscendingBeyondThisStart="";
                potentialfurtherAscendingBeyondThisEnd="";
                }
                else  //the previous number is too wide and need to write it as a standalone
                {
                    System.out.println("INSIDE HERE!!!!");
                    standaloneTemp = start;
```


```
[4.1->3.8, 3.9]
```

I could see it has missed this section

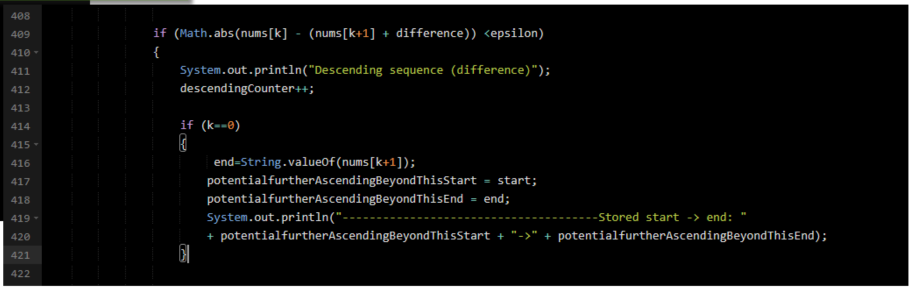

```
4.1f,4.0f,3.9f,3.8f,3.8f,3.7f,3.6f,3.9f
```

I checked through the logs and it became clearly evident.

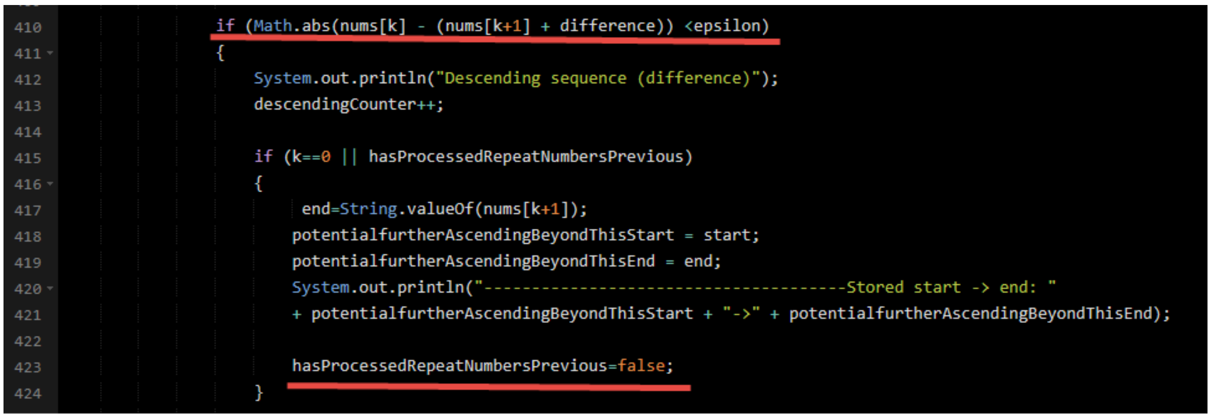


```
--------------------23229USING STORED TO WRITE RANGE
10101010Writing range: 4.1-> 3.8
asc counter: 0
desc counter: 3
test: 3
3
8
false

CHECKING: 3.8 with 3.7


currently in list: [4.1->3.8]
Descending sequence (difference)

CHECKING: 3.7 with 3.6


currently in list: [4.1->3.8]
Descending sequence (difference)

CHECKING: 3.6 with 3.9
```

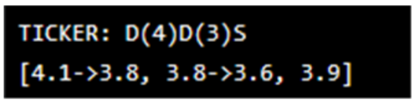
My code only faciliates
creating store if k==0
I need to make an exception if
it has come out of the main
else loop. But then I also need
to set the variable back to
initial state



```java
if (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon)
{
    System.out.println("Descending sequence (difference)");
    descendingCounter++;

    if (k==0)
    {
         end=String.valueOf(nums[k+1]);
        potentialfurtherAscendingBeyondThisStart = start;
        potentialfurtherAscendingBeyondThisEnd = end;
        System.out.println("------------------------------------Stored start -> end: "
        + potentialfurtherAscendingBeyondThisStart + "->" + potentialfurtherAscendingBeyondThisEnd);
    }

```

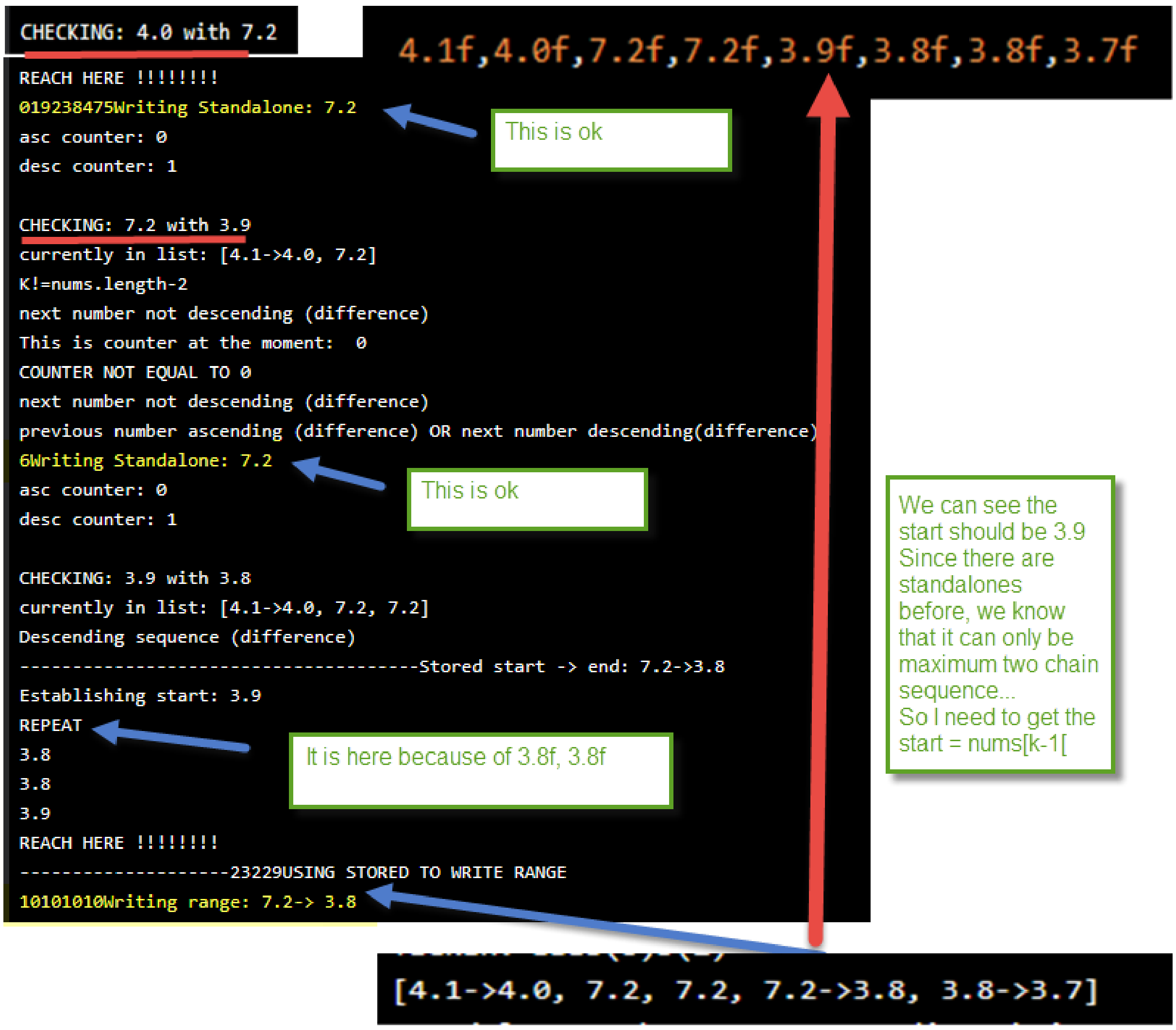


```java
if (Math.abs(nums[k] - (nums[k+1] + difference)) <epsilon)
{
    System.out.println("Descending sequence (difference)");
    descendingCounter++;

    if (k==0 || hasProcessedRepeatNumbersPrevious)
    {
         end=String.valueOf(nums[k+1]);
        potentialfurtherAscendingBeyondThisStart = start;
        potentialfurtherAscendingBeyondThisEnd = end;
        System.out.println("--------------------------------------Stored start -> end: "
        + potentialfurtherAscendingBeyondThisStart + "->" + potentialfurtherAscendingBeyondThisEnd);

        hasProcessedRepeatNumbersPrevious=false;
    }
```



```
TICKER: D(4)D(3)S
[4.1->3.8, 3.8->3.6, 3.9]
```

Just to be sure logic is correct, I have to try and experiment with standalone numbers in various positions..



```
CHECKING: 4.0 with 7.2
REACH HERE !!!!!!!!
019238475Writing Standalone: 7.2
asc counter: 0
desc counter: 1

CHECKING: 7.2 with 3.9
currently in list: [4.1->4.0, 7.2]
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  0
COUNTER NOT EQUAL TO 0
next number not descending (difference)
previous number ascending (difference) OR next number descending(difference)
6Writing Standalone: 7.2
asc counter: 0
desc counter: 1

CHECKING: 3.9 with 3.8
currently in list: [4.1->4.0, 7.2, 7.2]
Descending sequence (difference)
--------------------------------------Stored start -> end: 7.2->3.8
Establishing start: 3.9
REPEAT
3.8
3.8
3.9
REACH HERE !!!!!!!!
--------------------23229USING STORED TO WRITE RANGE
10101010Writing range: 7.2-> 3.8
```

```
4.1f,4.0f,7.2f,7.2f,3.9f,3.8f,3.8f,3.7f
```

This is ok

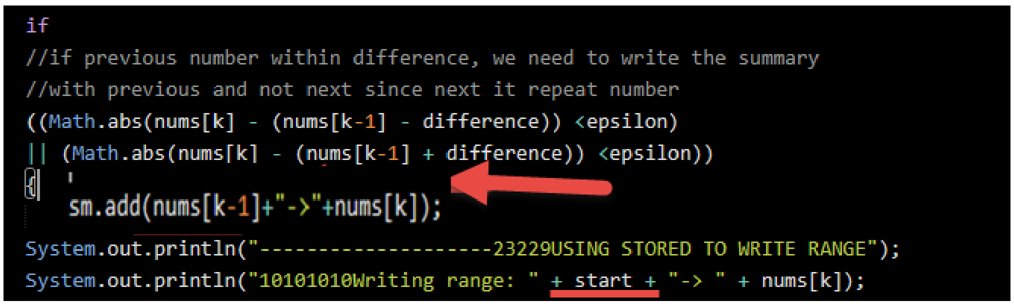
This is ok

It is here because of 3.8f, 3.8f

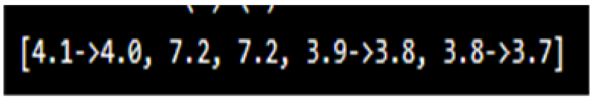
We can see the start should be 3.9 Since there are standalones before, we know that it can only be maximum two chain sequence...
So I need to get the start = nums[k-1[

```
[4.1->4.0, 7.2, 7.2, 7.2->3.8, 3.8->3.7]
```

I completed following change:

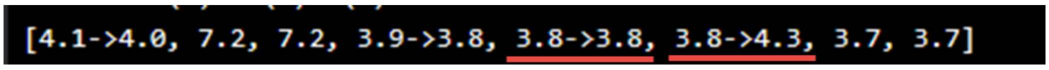


```
if
//if previous number within difference, we need to write the summary
//with previous and not next since next it repeat number
((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
|| (Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
{
    sm.add(nums[k-1]+"->"+nums[k]);
System.out.println("--------------------23229USING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

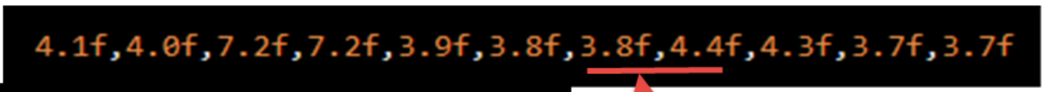


```
[4.1->4.0, 7.2, 7.2, 3.9->3.8, 3.8->3.7]
```

I am now going to try few more to ensure it is robust

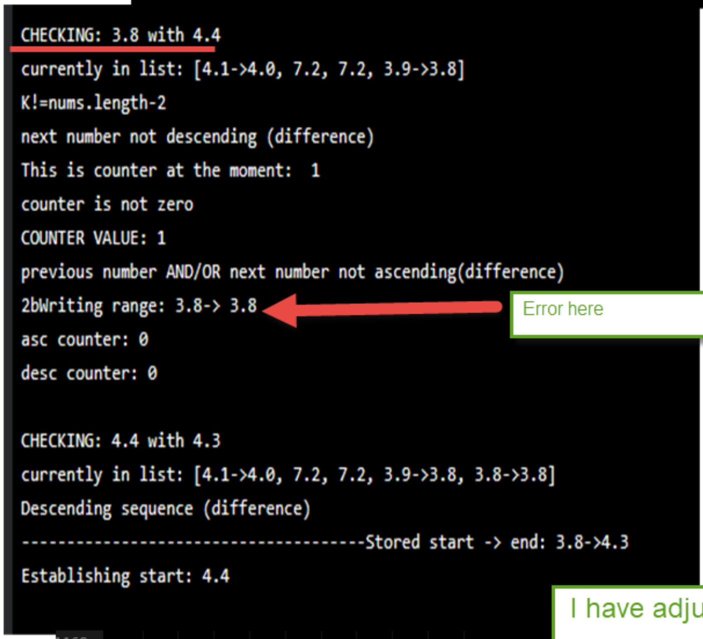


```
[4.1->4.0, 7.2, 7.2, 3.9->3.8, 3.8->3.8, 3.8->4.3, 3.7, 3.7]
```



```
4.1f,4.0f,7.2f,7.2f,3.9f,3.8f,3.8f,4.4f,4.3f,3.7f,3.7f
```

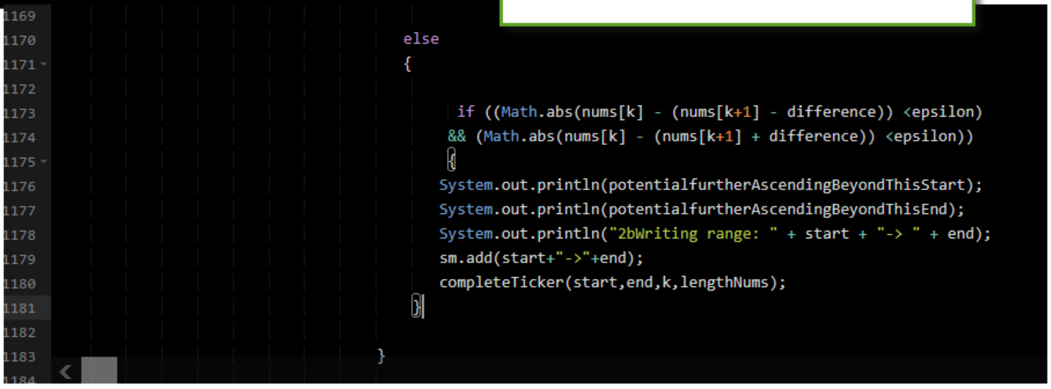
Expecting nothing here



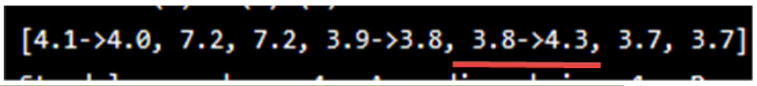
```
CHECKING: 3.8 with 4.4
currently in list: [4.1->4.0, 7.2, 7.2, 3.9->3.8]
K!=nums.length-2
next number not descending (difference)
This is counter at the moment:  1
counter is not zero
COUNTER VALUE: 1
previous number AND/OR next number not ascending(difference)
2bWriting range: 3.8-> 3.8
asc counter: 0
desc counter: 0

CHECKING: 4.4 with 4.3
currently in list: [4.1->4.0, 7.2, 7.2, 3.9->3.8, 3.8->3.8]
Descending sequence (difference)
-----------------------------------Stored start -> end: 3.8->4.3
Establishing start: 4.4
```

Error here

I have adjusted the code as below

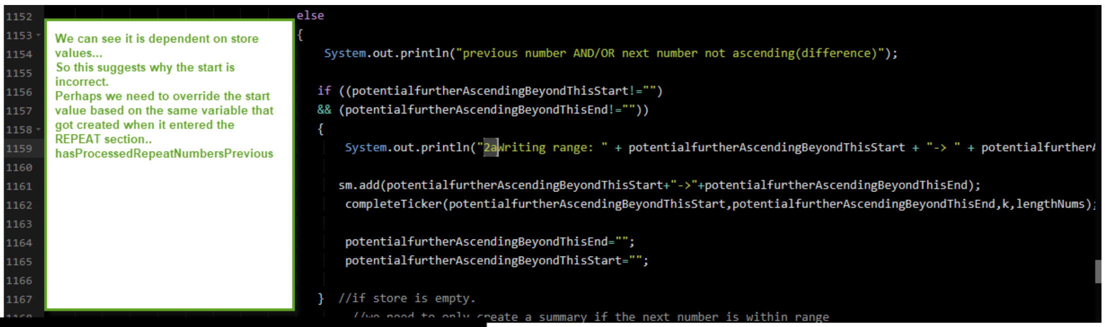


```
                else
                {

                    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);
                   }


                }
```



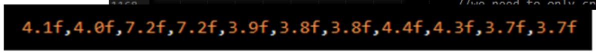
```
[4.1->4.0, 7.2, 7.2, 3.9->3.8, 3.8->4.3, 3.7, 3.7]
```

I now have one issue here



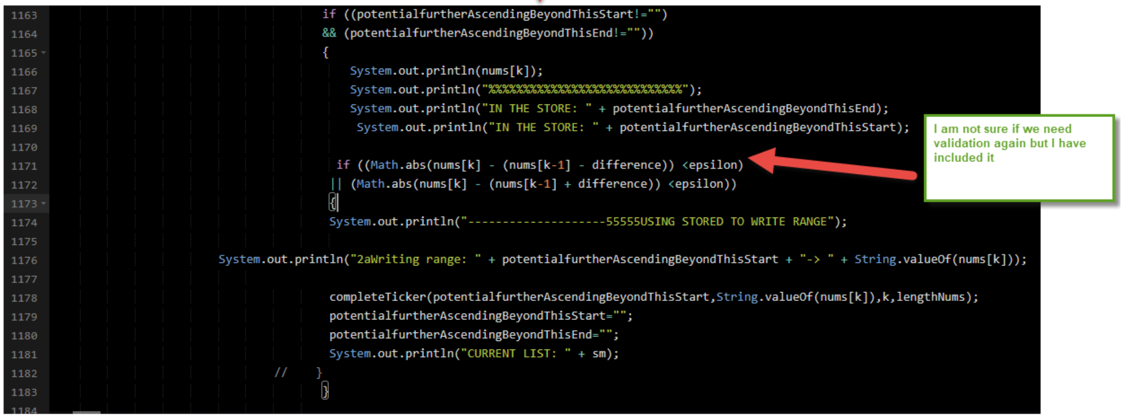
We can see it is dependent on store values...
So this suggests why the start is incorrect.
Perhaps we need to override the start value based on the same variable that got created when it entered the REPEAT section..
hasProcessedRepeatNumbersPrevious

```java
else
{
  System.out.println("previous number AND/OR next number not ascending(difference)");

 if ((potentialfurtherAscendingBeyondThisStart!="")
 && (potentialfurtherAscendingBeyondThisEnd!=""))
 {
    System.out.println("2aWriting range: " + potentialfurtherAscendingBeyondThisStart + "-> " + potentialfurther

   sm.add(potentialfurtherAscendingBeyondThisStart+"->"+potentialfurtherAscendingBeyondThisEnd);
    completeTicker(potentialfurtherAscendingBeyondThisStart,potentialfurtherAscendingBeyondThisEnd,k,lengthNums);

    potentialfurtherAscendingBeyondThisEnd="";
    potentialfurtherAscendingBeyondThisStart="";

 }  //if store is empty.
```



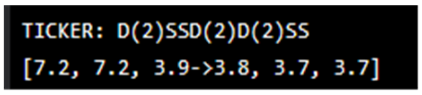
```
4.1f,4.0f,7.2f,7.2f,3.9f,3.8f,3.8f,4.4f,4.3f,3.7f,3.7f
```

NEW CODE



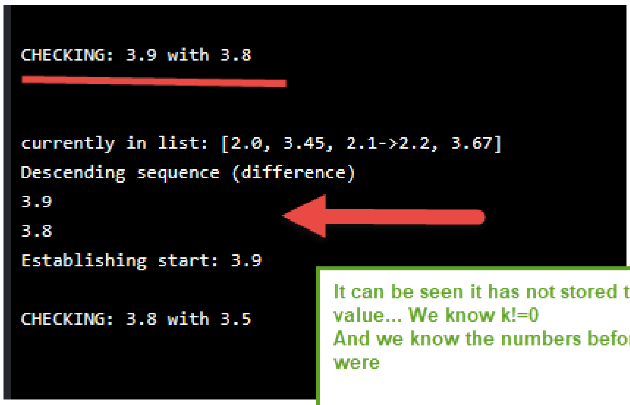
```java
                              if ((potentialfurtherAscendingBeyondThisStart!="")
                              && (potentialfurtherAscendingBeyondThisEnd!=""))
                              {
                                  System.out.println(nums[k]);
                                  System.out.println("%%%%%%%%%%%%%%%%%%%%%%%%%%%%");
                                  System.out.println("IN THE STORE: " + potentialfurtherAscendingBeyondThisEnd);
                                   System.out.println("IN THE STORE: " + potentialfurtherAscendingBeyondThisStart);

                                if ((Math.abs(nums[k] - (nums[k-1] - difference)) <epsilon)
                               || (Math.abs(nums[k] - (nums[k-1] + difference)) <epsilon))
                               {
                               System.out.println("-------------------55555USING STORED TO WRITE RANGE");

               System.out.println("2aWriting range: " + potentialfurtherAscendingBeyondThisStart + "-> " + String.valueOf(nums[k]));

                               completeTicker(potentialfurtherAscendingBeyondThisStart,String.valueOf(nums[k]),k,lengthNums);
                               potentialfurtherAscendingBeyondThisStart="";
                               potentialfurtherAscendingBeyondThisEnd="";
                               System.out.println("CURRENT LIST: " + sm);
                          //    }
                              }
```

I am not sure if we need validation again but I have included it



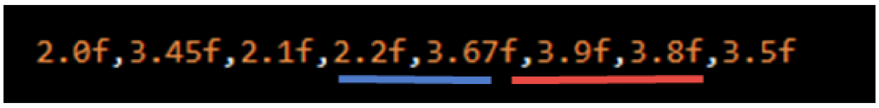
```
TICKER: D(2)SSD(2)D(2)SS
[7.2, 7.2, 3.9->3.8, 3.7, 3.7]
```

I will just complete all my test cases again.. It is difficult to keep getting more sophisticated with standalones but I will try few more...



```
CHECKING: 3.9 with 3.8

currently in list: [2.0, 3.45, 2.1->2.2, 3.67]
Descending sequence (difference)
3.9
3.8
Establishing start: 3.9

CHECKING: 3.8 with 3.5
```

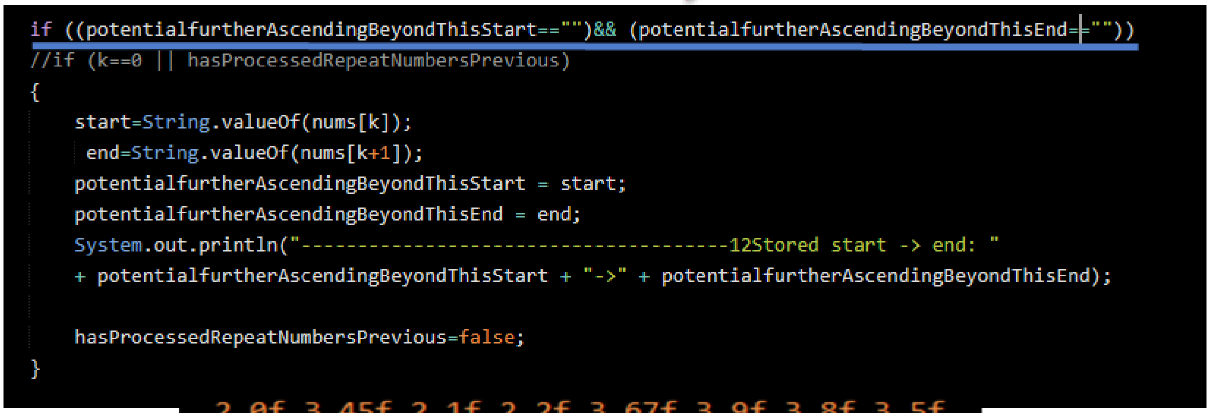
It can be seen it has not stored the value... We know k!=0
And we know the numbers before were



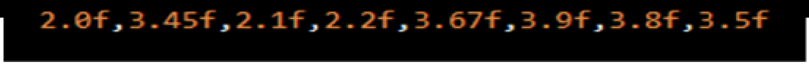
```
2.0f,3.45f,2.1f,2.2f,3.67f,3.9f,3.8f,3.5f
```

Once again, I do not want to look too far back in the sequence since this is concerned mainly with adjacent numbers..

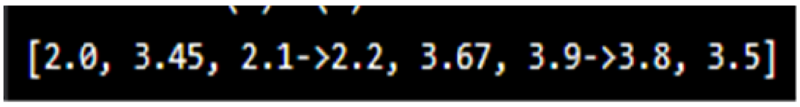
I have chosen the most obvious scenario



```
if ((potentialfurtherAscendingBeyondThisStart=="")&& (potentialfurtherAscendingBeyondThisEnd==""))
//if (k==0 || hasProcessedRepeatNumbersPrevious)
{
    start=String.valueOf(nums[k]);
     end=String.valueOf(nums[k+1]);
    potentialfurtherAscendingBeyondThisStart = start;
    potentialfurtherAscendingBeyondThisEnd = end;
    System.out.println("-------------------------------------12Stored start -> end: "
    + potentialfurtherAscendingBeyondThisStart + "->" + potentialfurtherAscendingBeyondThisEnd);

    hasProcessedRepeatNumbersPrevious=false;
}
```



```
2.0f,3.45f,2.1f,2.2f,3.67f,3.9f,3.8f,3.5f
```



```
[2.0, 3.45, 2.1->2.2, 3.67, 3.9->3.8, 3.5]
```

It is time to give up... It is too difficult