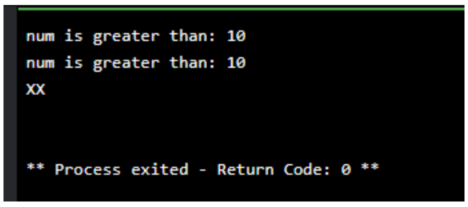
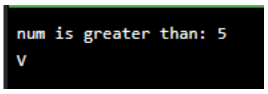
This is my formal testing..
I am going to start at most trivial level....

TEST CASE 1:   Decimal input 20   PASS



```
num is greater than: 10
num is greater than: 10
XX


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

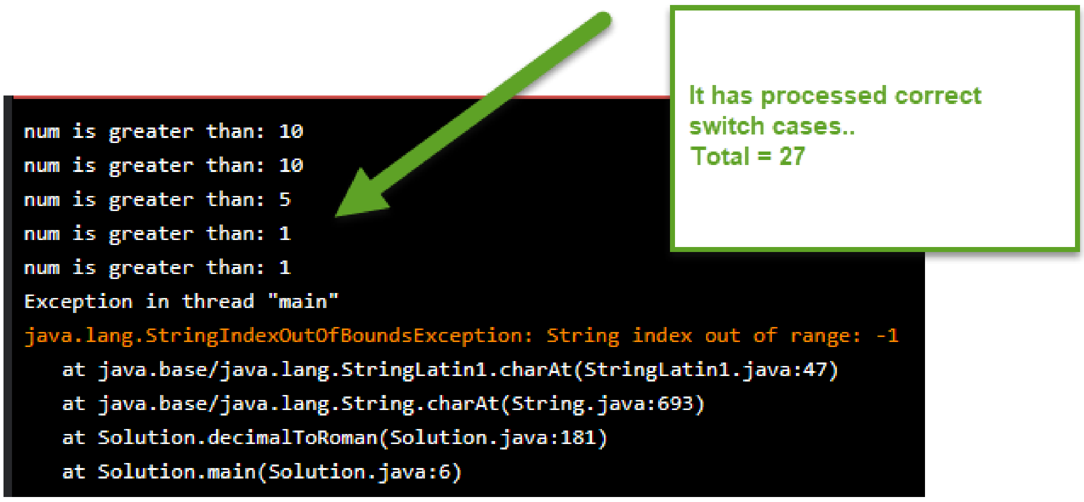
TEST CASE 2:   Decimal input 5   PASS



```
num is greater than: 5
V
```

TEST CASE 3:   Decimal 27   PASS

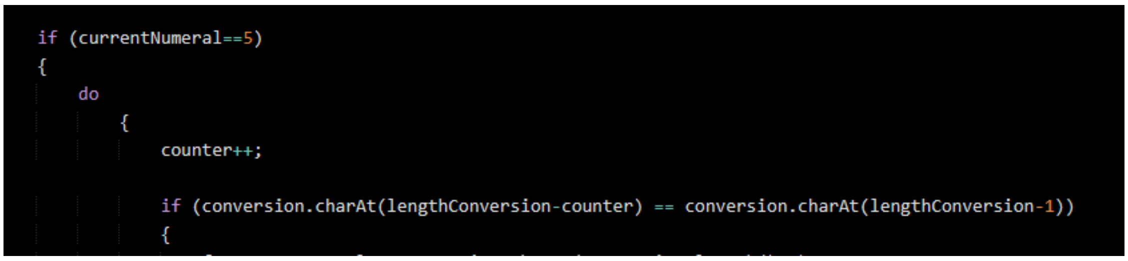
It now appears much easier to troubleshoot with references to Line 47 and Line 6 and Line 181



```
num is greater than: 10
num is greater than: 10
num is greater than: 5
num is greater than: 1
num is greater than: 1
Exception in thread "main"
java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:47)
    at java.base/java.lang.String.charAt(String.java:693)
    at Solution.decimalToRoman(Solution.java:181)
    at Solution.main(Solution.java:6)
```

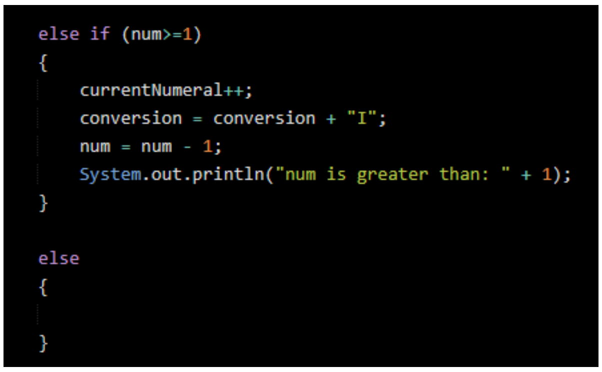
I will mass remediate issues since it is an exercise I wish to conclude at earliest opportunity.

But I can see that it entered in here on basis currentNumeral==5.
It was only intended if there were consecutive repeat numerals.



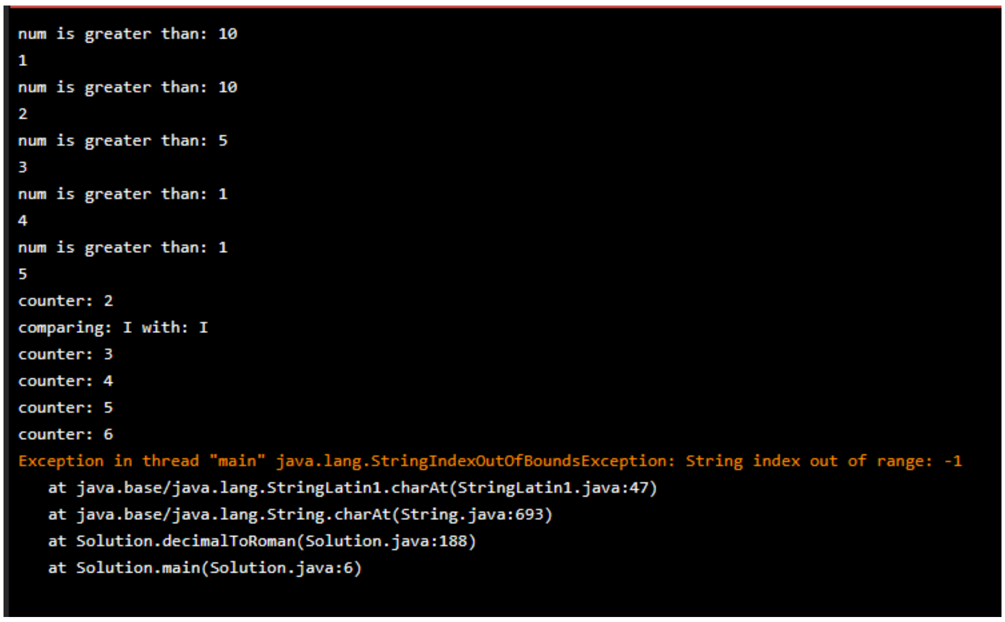
```
if (currentNumeral==5)
{
    do
        {
            counter++;

            if (conversion.charAt(lengthConversion-counter) == conversion.charAt(lengthConversion-1))
            {
```

It has been identified to be issue in my else statement in the if-else.
I have created a separate equivalent to default in Switch statement and will execute again.



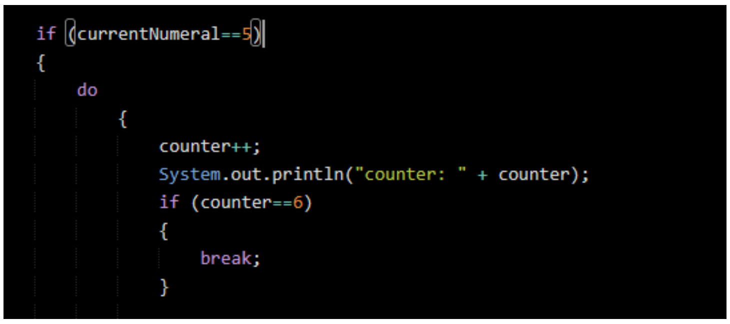
```
else if (num>=1)
{
    currentNumeral++;
    conversion = conversion + "I";
    num = num - 1;
    System.out.println("num is greater than: " + 1);
}

else
{

}
```

I further experienced...

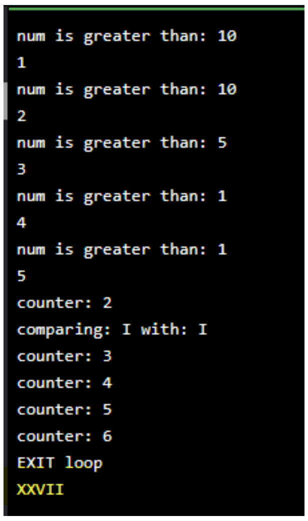


```
num is greater than: 10
1
num is greater than: 10
2
num is greater than: 5
3
num is greater than: 1
4
num is greater than: 1
5
counter: 2
comparing: I with: I
counter: 3
counter: 4
counter: 5
counter: 6
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.base/java.lang.StringLatin1.charAt(StringLatin1.java:47)
    at java.base/java.lang.String.charAt(String.java:693)
    at Solution.decimalToRoman(Solution.java:188)
    at Solution.main(Solution.java:6)
```

I knew straight away that counter is potentially too high, so introduced few more changes as below:


```
if ((currentNumeral==5))
{
    do
    {
        counter++;
        System.out.println("counter: " + counter);
        if (counter==6)
        {
            break;
        }
```

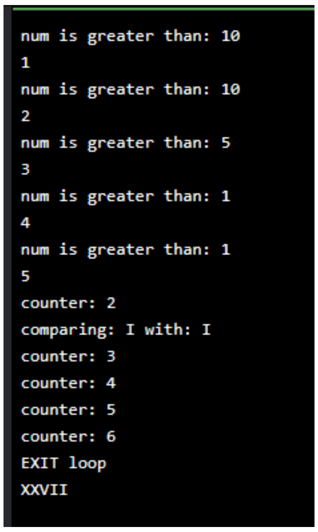
TEST CASE 3a: 27 retested


```
num is greater than: 10
1
num is greater than: 10
2
num is greater than: 5
3
num is greater than: 1
4
num is greater than: 1
5
counter: 2
comparing: I with: I
counter: 3
counter: 4
counter: 5
counter: 6
EXIT loop
XXVII
```

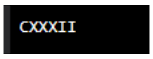
Before I start exploring any cases in which it will trigger four consecutive numerals such as:
 94 (LXXXXIIII), this will be a complex test since it will perform adjustment two fold..
 42 (XXXXII) – this seems a more sensible approach….

I will try few more basic scenarios with higher decimals…

TEST CASE 4:   27   retested   PASS

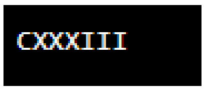

```
num is greater than: 10
1
num is greater than: 10
2
num is greater than: 5
3
num is greater than: 1
4
num is greater than: 1
5
counter: 2
comparing: I with: I
counter: 3
counter: 4
counter: 5
counter: 6
EXIT loop
XXVII
```

TEST CASE 5: 132 PASS



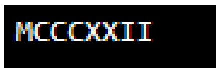
```
CXXXII
```

TEST CASE 6: 133 PASS



```
CXXXIII
```

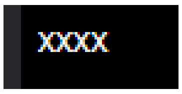
TEST CASE 7: 1322 PASS



```
MCCCXXII
```

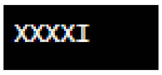
So far I have every reason to believe that the code is comfortable, I will now explore 40, 41, 42, 43
Only I feel comfortable, I will explore up to 50
I have reason to believe that once all these numbers function, it will become a multi-functional solution

TEST CASE 8 : 40 FAIL



```
XXXX
```

TEST CASE 9: 41 FAIL



```
XXXXI
```

I could see it was not entering in the area where it should have remediated the XXXX
So I performed several small changes to my code.

```
if (currentNumeral==4)
      {
         do
           {
              counter++;
              System.out.println("counter: " + counter);

              if (counter==currentNumeral)
              {
                break;
              }


              if (conversion.charAt(lengthConversion-counter) ==
conversion.charAt(lengthConversion-1))
              {
                System.out.println("comparing: " + conversion.charAt(lengthConversion-
counter) + " with: " + conversion.charAt(lengthConversion-1));
                fourInRowNumeral = conversion.charAt(conversion.length()-1);
```

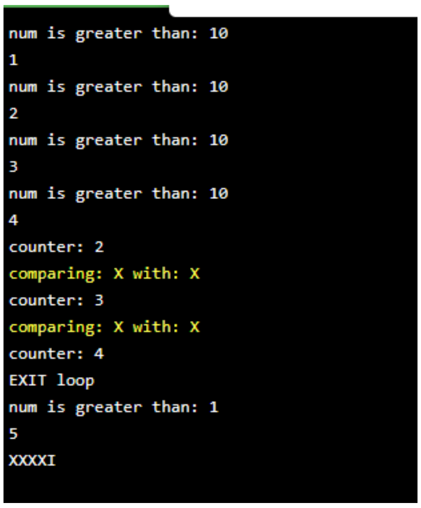
```
                if (currentNumeral>4)
                {
                incorrectPredecessor = conversion.charAt(conversion.length()-5);
                }
```

```
                match++;


              } //end of big if
```

```
                }while(counter<currentNumeral);
```

TEST CASE 9a: 41   FAIL

It is now interacting with expected area of code, I will keep focus on this.



```
num is greater than: 10
1
num is greater than: 10
2
num is greater than: 10
3
num is greater than: 10
4
counter: 2
comparing: X with: X
counter: 3
comparing: X with: X
counter: 4
EXIT loop
num is greater than: 1
5
XXXXI
```

TEST CASE 9b: 41   PASS

I have now added an aggressive amount of coding.
I will save this as version Test version 9b.
It is beginning to look much more like the finished article.
And I can safely say that being familiar with IDE has assisted, since my progression is much quicker.

```
num is greater than: 10

1

*************REMAINING num: 31

num is greater than: 10

2

*************REMAINING num: 21

num is greater than: 10

3

*************REMAINING num: 11

num is greater than: 10

4

*************REMAINING num: 1

counter: 1

comparing: X with: X

counter: 2

comparing: X with: X

counter: 3

comparing: X with: X

counter: 4

EXIT loop

THREE MATCHES FOUND

0000000000000000000000000000

REMEDIATING XXXX and IIII

---------------------CONVERSION: XXXX

CORRECT NUMERAL: XXXX

MATCH

Correction: XXXX=>XL

num is greater than: 1

5

*************REMAINING num: 0

counter: 5

EXIT loop

THREE MATCHES FOUND

0000000000000000000000000000

REMEDIATING XXXX and IIII

---------------------CONVERSION: XLI
```

```
CORRECT NUMERAL: XXXX

XLI



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

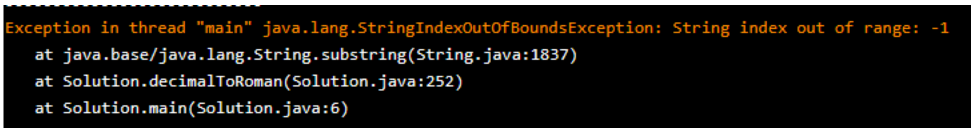
I will quickly try my earlier test cases to ensure no interruption...

TEST CASE 1:   20   => PASS
TEST CASE 2:   5   => PASS
TEST CASE 3:   27   => PASS
TEST CASE 5:   132   => PASS
TEST CASE 6:   133   => PASS
TEST CASE 7:   1322   => PASS
TEST CASE 8 :   40   =>   PASS
TEST CASE 9b: 41   =>   PASS

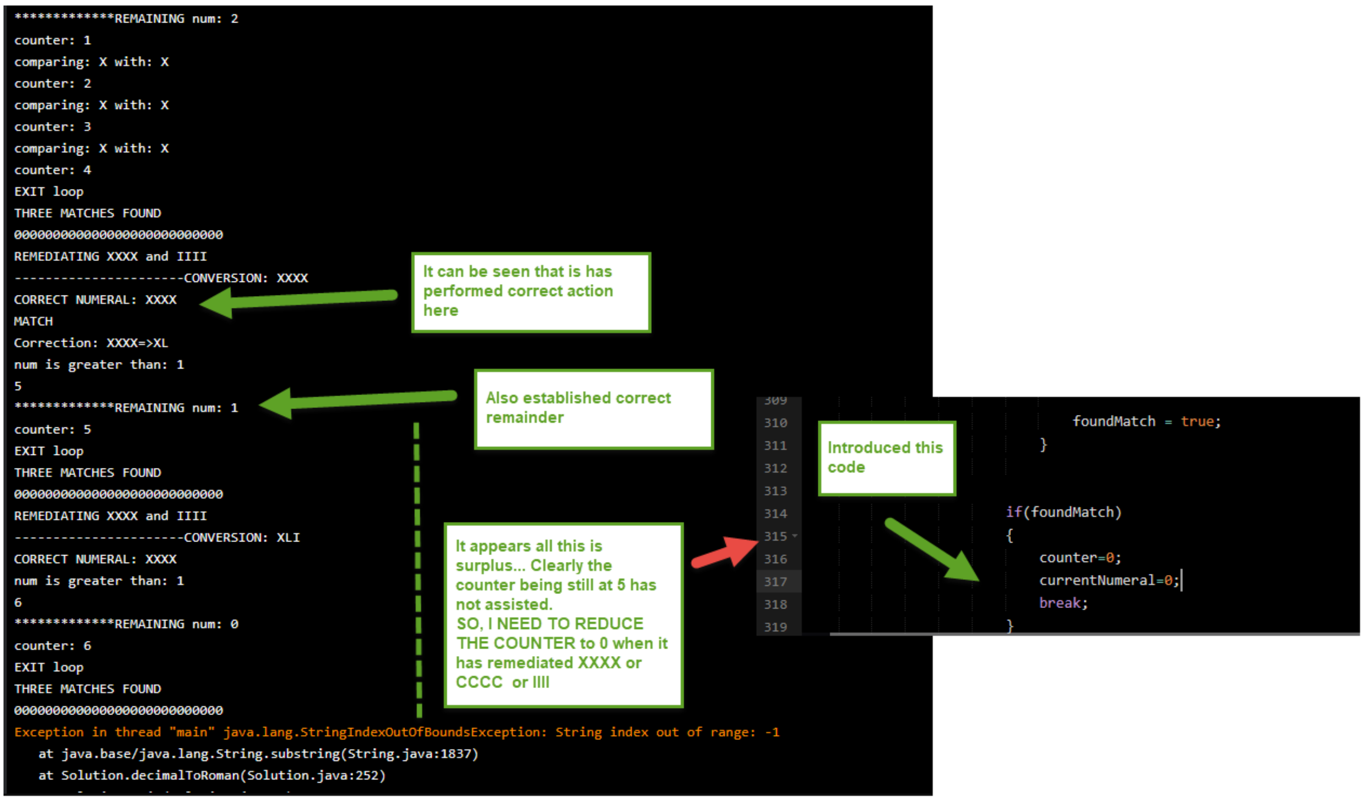
My logic suggests if I can progress from 42 => 50

I expect it to be complete.. The numbers which are unsettling are 44 (XXXXIIII) and 49 (XXXXVIIII).

TEST CASE 10 :   42   FAIL



```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.base/java.lang.String.substring(String.java:1837)
    at Solution.decimalToRoman(Solution.java:252)
    at Solution.main(Solution.java:6)
```



```
*************REMAINING num: 2
counter: 1
comparing: X with: X
counter: 2
comparing: X with: X
counter: 3
comparing: X with: X
counter: 4
EXIT loop
THREE MATCHES FOUND
000000000000000000000000000
REMEDIATING XXXX and IIII
---------------------CONVERSION: XXXX
CORRECT NUMERAL: XXXX
MATCH
Correction: XXXX=>XL
num is greater than: 1
5
*************REMAINING num: 1
counter: 5
EXIT loop
THREE MATCHES FOUND
000000000000000000000000000
REMEDIATING XXXX and IIII
---------------------CONVERSION: XLI
CORRECT NUMERAL: XXXX
num is greater than: 1
6
*************REMAINING num: 0
counter: 6
EXIT loop
THREE MATCHES FOUND
000000000000000000000000000
Exception in thread "main" java.lang.StringIndexOutOfBoundsException: String index out of range: -1
    at java.base/java.lang.String.substring(String.java:1837)
    at Solution.decimalToRoman(Solution.java:252)
```

It can be seen that is has performed correct action here

Also established correct remainder

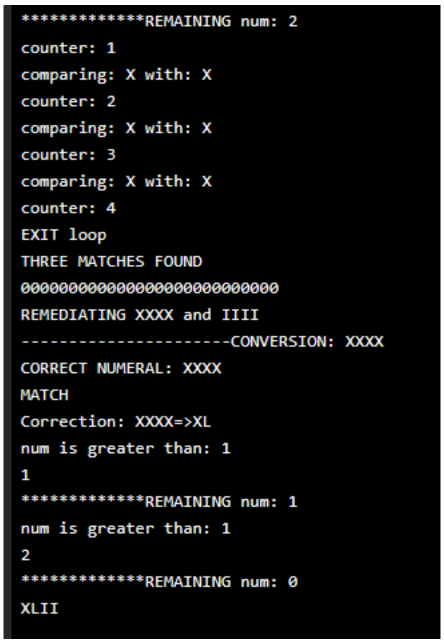
It appears all this is surplus... Clearly the counter being still at 5 has not assisted.
SO, I NEED TO REDUCE THE COUNTER to 0 when it has remediated XXXX or CCCC or IIII

Introduced this code

```
            foundMatch = true;
        }


    if(foundMatch)
    {
        counter=0;
        currentNumeral=0;
        break;
    }
```

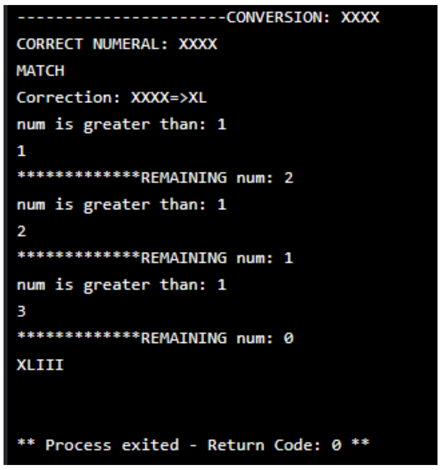
TEST CASE 10a: 42   PASS

It is looking much tidier



```
*************REMAINING num: 2
counter: 1
comparing: X with: X
counter: 2
comparing: X with: X
counter: 3
comparing: X with: X
counter: 4
EXIT loop
THREE MATCHES FOUND
000000000000000000000000000
REMEDIATING XXXX and IIII
----------------------CONVERSION: XXXX
CORRECT NUMERAL: XXXX
MATCH
Correction: XXXX=>XL
num is greater than: 1
1
*************REMAINING num: 1
num is greater than: 1
2
*************REMAINING num: 0
XLII
```

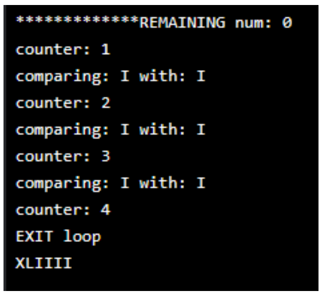
TEST CASE 11:   43   PASS



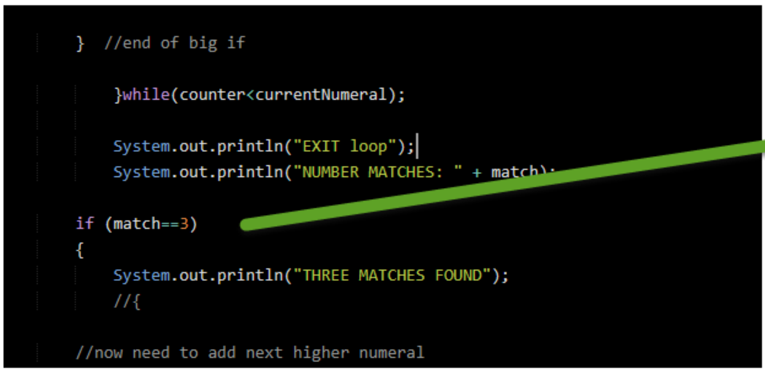
```
----------------------CONVERSION: XXXX
CORRECT NUMERAL: XXXX
MATCH
Correction: XXXX=>XL
num is greater than: 1
1
*************REMAINING num: 2
num is greater than: 1
2
*************REMAINING num: 1
num is greater than: 1
3
*************REMAINING num: 0
XLIII


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

TEST CASE 12:   44   = FAIL

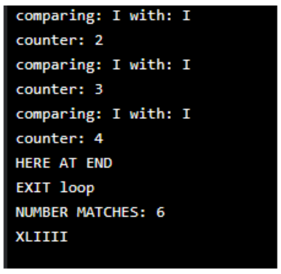
This is one of the scenarios which will test robustness of the code



```
*************REMAINING num: 0
counter: 1
comparing: I with: I
counter: 2
comparing: I with: I
counter: 3
comparing: I with: I
counter: 4
EXIT loop
XLIIII
```

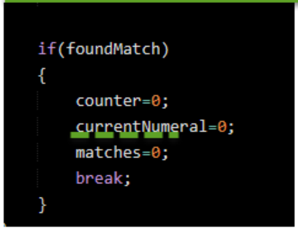


```
    }  //end of big if

        }while(counter<currentNumeral);

        System.out.println("EXIT loop");
        System.out.println("NUMBER MATCHES: " + match);

    if (match==3)
    {
        System.out.println("THREE MATCHES FOUND");
        //{

    //now need to add next higher numeral
```



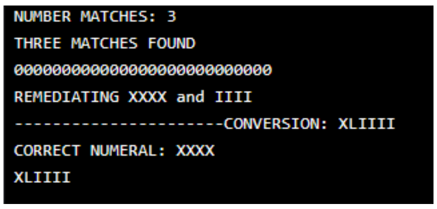
```
comparing: I with: I
counter: 2
comparing: I with: I
counter: 3
comparing: I with: I
counter: 4
HERE AT END
EXIT loop
NUMBER MATCHES: 6
XLIIII
```

I immediately realised I had not performed the following operation:



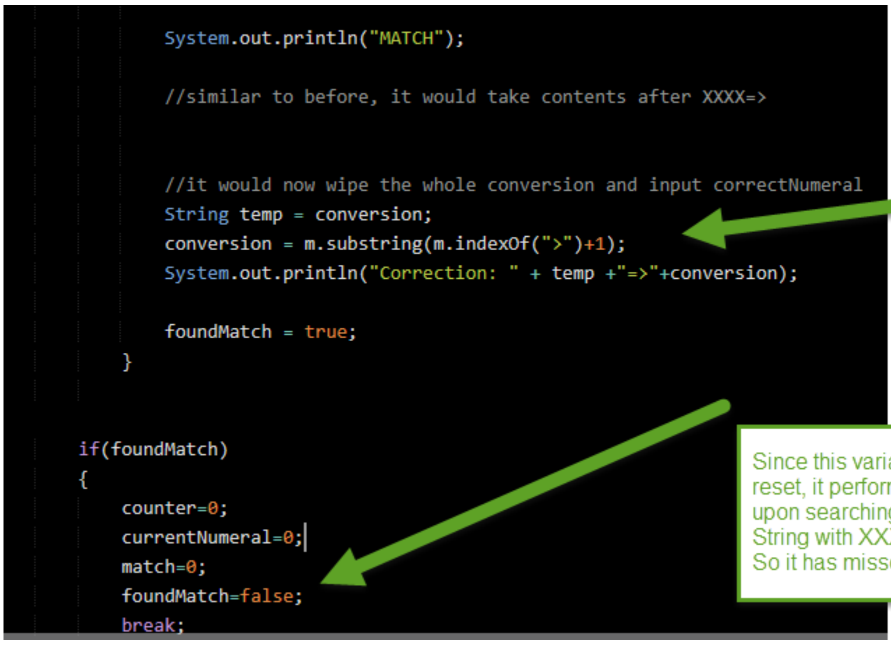
```
if(foundMatch)
{
    counter=0;
    currentNumeral=0;
    matches=0;
    break;
}
```

<u>TEST CASE 12a: 44 FAIL</u>



```
NUMBER MATCHES: 3
THREE MATCHES FOUND
00000000000000000000000000
REMEDIATING XXXX and IIII
---------------------CONVERSION: XLIIII
CORRECT NUMERAL: XXXX
XLIIII
```

But it is making lot more sense now, clearly it is comparing XLIIII
But it did not enter the loop to identify a successful match because the following variable was not reset:

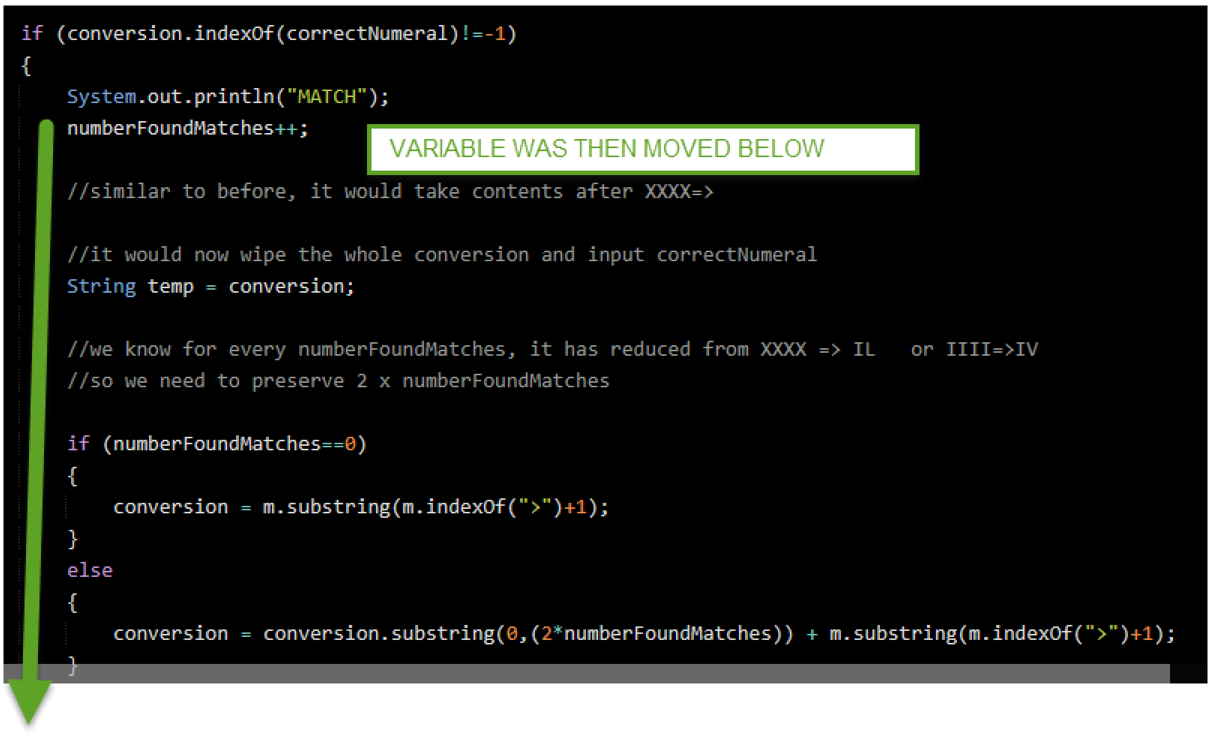


```
            System.out.println("MATCH");

            //similar to before, it would take contents after XXXX=>


            //it would now wipe the whole conversion and input correctNumeral
            String temp = conversion;
            conversion = m.substring(m.indexOf(">")+1);
            System.out.println("Correction: " + temp +"=>"+conversion);

            foundMatch = true;
        }

    if(foundMatch)
    {
        counter=0;
        currentNumeral=0;
        match=0;
        foundMatch=false;
        break;
```

This is the next issue, it is currently designed to wipe the entire conversion.. I can simply adjust this so that it keeps everything before the last 4 numerals in conversion

Since this variable was not reset, it performed a break upon searching Conversion String with XXXX
So it has missed out IIII and

This was the next issue, since as can be seen above, the conversion string will be overwritten. This was not an issue when XXXX => IL
But it was unaware of further remaining decimal.
I implemented the following:



```
if (conversion.indexOf(correctNumeral)!=-1)
{
    System.out.println("MATCH");
    numberFoundMatches++;

    //similar to before, it would take contents after XXXX=>

    //it would now wipe the whole conversion and input correctNumeral
    String temp = conversion;

    //we know for every numberFoundMatches, it has reduced from XXXX => IL   or IIII=>IV
    //so we need to preserve 2 x numberFoundMatches

    if (numberFoundMatches==0)
    {
        conversion = m.substring(m.indexOf(">")+1);
    }
    else
    {
        conversion = conversion.substring(0,(2*numberFoundMatches)) + m.substring(m.indexOf(">")+1);
    }
```

TEST CASE 12b: 44 PASS

```
num is greater than: 10

1

*************REMAINING num: 34

num is greater than: 10

2

*************REMAINING num: 24

num is greater than: 10

3

*************REMAINING num: 14

num is greater than: 10

4

*************REMAINING num: 4

counter: 1

comparing: X with: X

counter: 2

comparing: X with: X

counter: 3

comparing: X with: X

counter: 4

HERE AT END

EXIT loop

NUMBER MATCHES: 3

THREE MATCHES FOUND

000000000000000000000000000
```

```
REMEDIATING XXXX and IIII and CCCC

----------------------ENTRY IN CONVERTED STRING: XXXX

INCORRECT NUMERAL: XXXX

MATCH FOUND IN EXISTING CONVERSION STRING
```

Correction: XXXX=>XL

```
num is greater than: 1

1

*************REMAINING num: 3

num is greater than: 1

2

*************REMAINING num: 2

num is greater than: 1

3

*************REMAINING num: 1

num is greater than: 1

4

*************REMAINING num: 0

counter: 1

comparing: I with: I

counter: 2

comparing: I with: I

counter: 3

comparing: I with: I

counter: 4

HERE AT END

EXIT loop

NUMBER MATCHES: 3

THREE MATCHES FOUND

000000000000000000000000000

REMEDIATING XXXX and IIII and CCCC

----------------------ENTRY IN CONVERTED STRING: XLIIII

INCORRECT NUMERAL: XXXX

----------------------ENTRY IN CONVERTED STRING: XLIIII

INCORRECT NUMERAL: IIII

MATCH FOUND IN EXISTING CONVERSION STRING
```

Correction: XLIIII=>XLIV

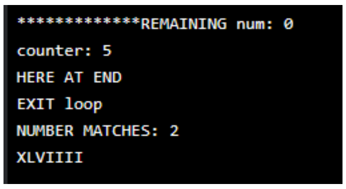
XLIV

** Process exited - Return Code: 0 **

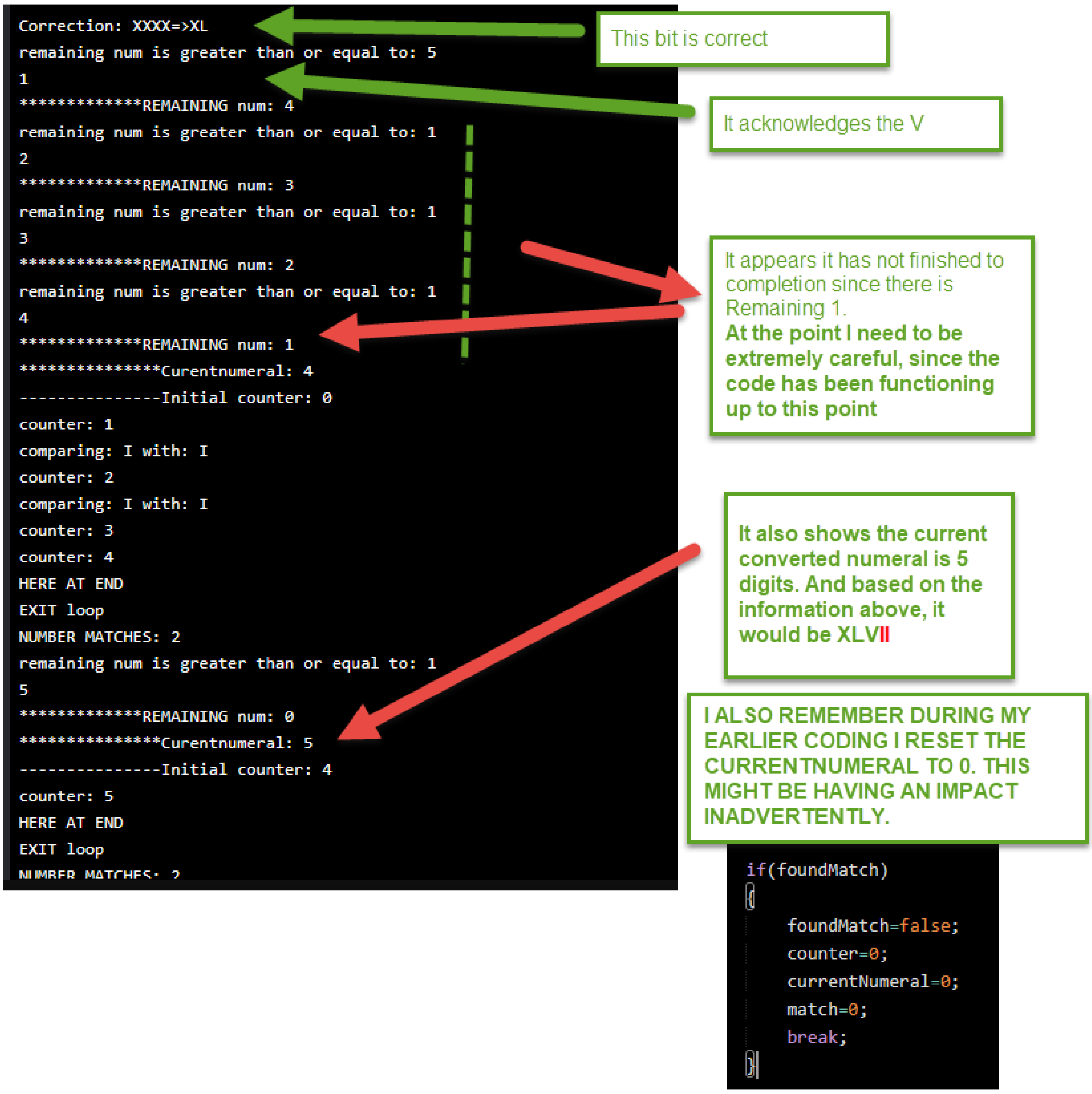
Now I need to continue testing from 45 => 50

TEST CASE 13:   45 => 48 (PASS)

TEST CASE 14:   49 = FAIL



```
*************REMAINING num: 0
counter: 5
HERE AT END
EXIT loop
NUMBER MATCHES: 2
XLVIIII
```



```
Correction: XXXX=>XL
remaining num is greater than or equal to: 5
1
*************REMAINING num: 4
remaining num is greater than or equal to: 1
2
*************REMAINING num: 3
remaining num is greater than or equal to: 1
3
*************REMAINING num: 2
remaining num is greater than or equal to: 1
4
*************REMAINING num: 1
***************Curentnumeral: 4
---------------Initial counter: 0
counter: 1
comparing: I with: I
counter: 2
comparing: I with: I
counter: 3
counter: 4
HERE AT END
EXIT loop
NUMBER MATCHES: 2
remaining num is greater than or equal to: 1
5
*************REMAINING num: 0
***************Curentnumeral: 5
---------------Initial counter: 4
counter: 5
HERE AT END
EXIT loop
NUMBER MATCHES: 2
```

This bit is correct

It acknowledges the V

It appears it has not finished to completion since there is Remaining 1.
At the point I need to be extremely careful, since the code has been functioning up to this point

It also shows the current converted numeral is 5 digits. And based on the information above, it would be XLVII

I ALSO REMEMBER DURING MY EARLIER CODING I RESET THE CURRENTNUMERAL TO 0. THIS MIGHT BE HAVING AN IMPACT INADVERTENTLY.

```
if(foundMatch)
{
    foundMatch=false;
    counter=0;
    currentNumeral=0;
    match=0;
    break;
}
```

I am also finding that is has not passed decimal 9
I think this is a better starting point...
It suggests to me it is related to having a V in front... And this is affecting the counter along the way. I think it will be investigate with this first..

TEST CASE 15: 9 FAIL

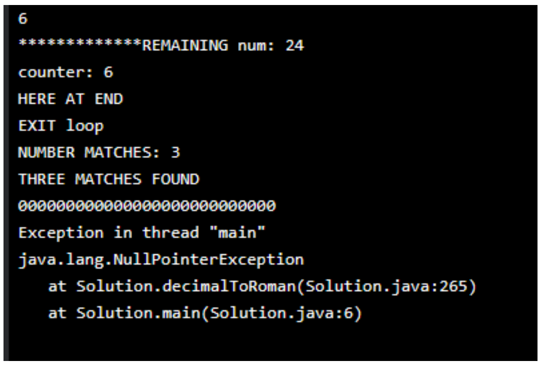
```
remaining num is greater than or equal to: 5

1

*************REMAINING num: 4

remaining num is greater than or equal to: 1

2

*************REMAINING num: 3

remaining num is greater than or equal to: 1

3

*************REMAINING num: 2

remaining num is greater than or equal to: 1

4

*************REMAINING num: 1

***************Curentnumeral: 4

---------------Initial counter: 0

counter: 1

------------------------------------CONVERSION: VIII    //The problem arises here straight away.

                                                        //There is something in my logic forcing below action as soon
                                                        as currentNumeral in conversion is 4

comparing: I with: I

counter: 2

------------------------------------CONVERSION: VIII

comparing: I with: I

counter: 3

------------------------------------CONVERSION: VIII

counter: 4

HERE AT END

EXIT loop

NUMBER MATCHES: 2

remaining num is greater than or equal to: 1

5

*************REMAINING num: 0
```

```
***************Curentnumeral: 5

---------------Initial counter: 4

counter: 5

HERE AT END

EXIT loop

NUMBER MATCHES: 2

VIIII

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

Now it is a case of trying more sequences with CCCCXXXXIIII
This seems like a perfect test

<u>TEST CASE 16:   4044   = FAIL</u>

I did feel it was too adventurous. My next logical tests beyond



```
6
*************REMAINING num: 24
counter: 6
HERE AT END
EXIT loop
NUMBER MATCHES: 3
THREE MATCHES FOUND
000000000000000000000000000
Exception in thread "main"
java.lang.NullPointerException
    at Solution.decimalToRoman(Solution.java:265)
    at Solution.main(Solution.java:6)
```

<u>This appears again the more I fix, it will have implications elsewhere.</u>

The easiest solution now is to let it convert it naturally.

And then replace all CCCC with ID
XXXX => XL
IIII => IV

This will solve all the headache completely and simplify the problem.