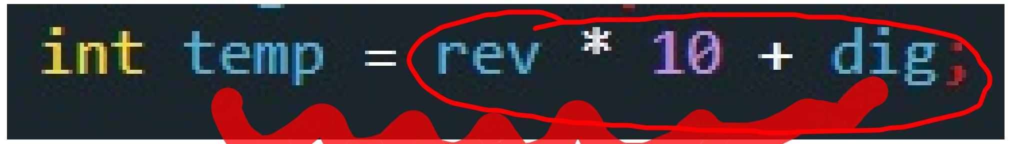
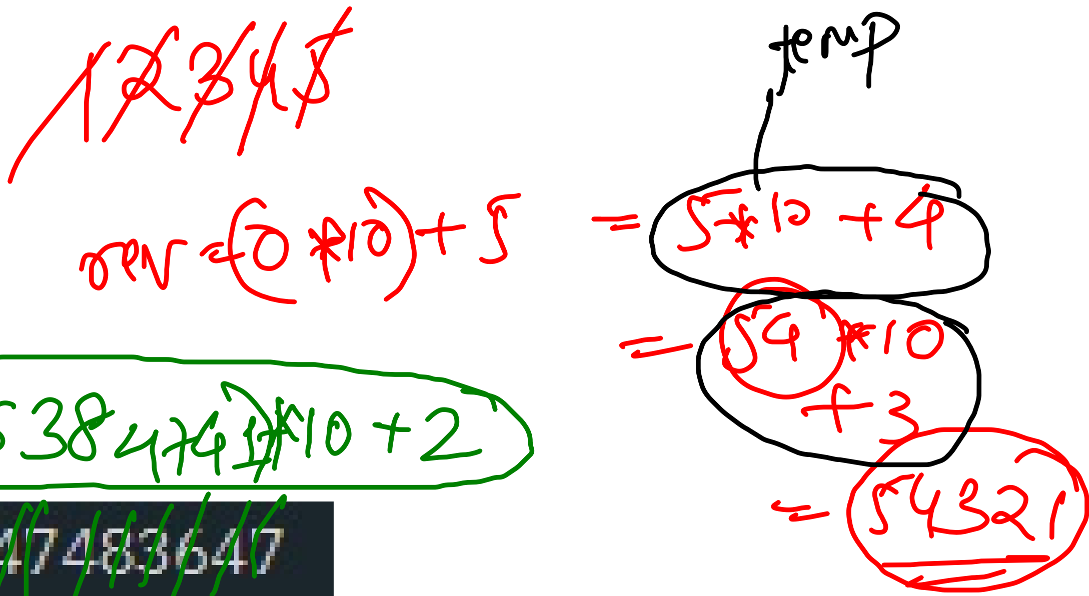
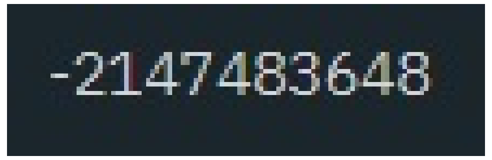
Reverse Integer



```
int temp = rev * 10 + dig;
```

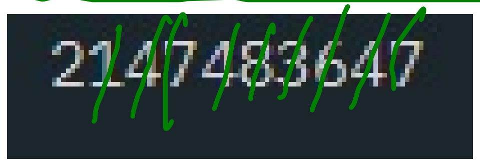


string ← Integer → long



```
-2147483648
```

$<$ Integer $<$



```
2147483647
```

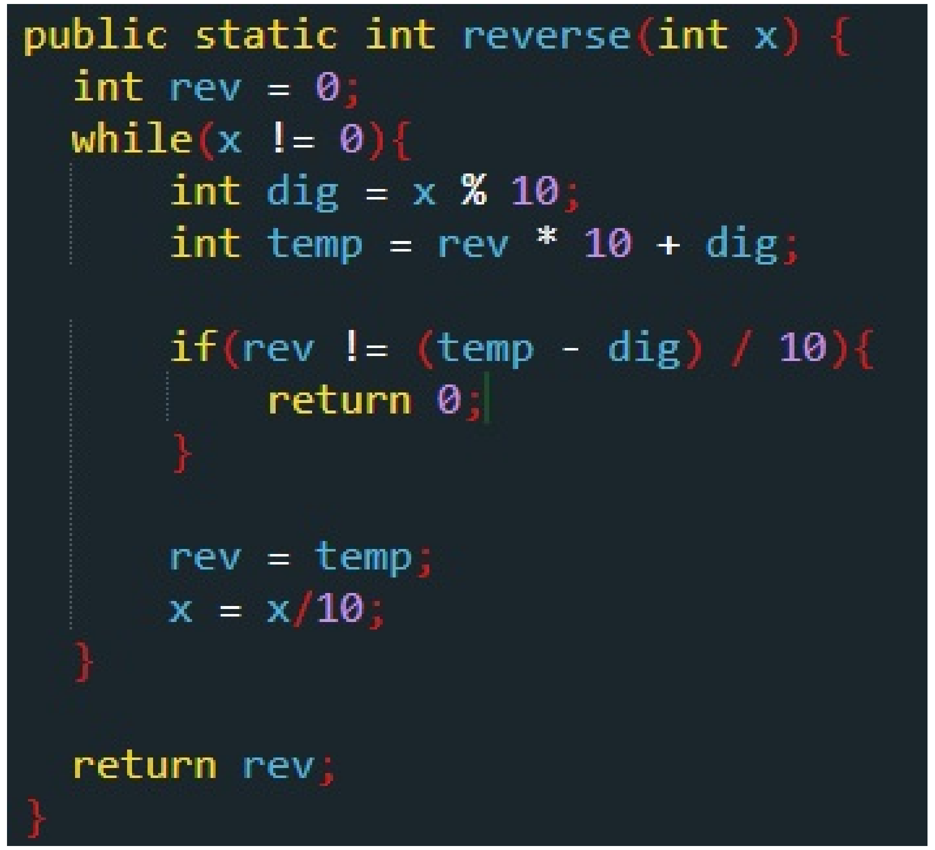
−Infinity
{Integer.MIN_VALUE}
$\Rightarrow \{-2^{31}\}$

+infinity
{Integer.MAX_VALUE}
$\Rightarrow \{2^{31}-1\}$

$$\boxed{\text{Rev} \;!= \frac{\text{temp} - \text{digit}}{10}}$$

↳ temp overflow ⇒ return 0

Code



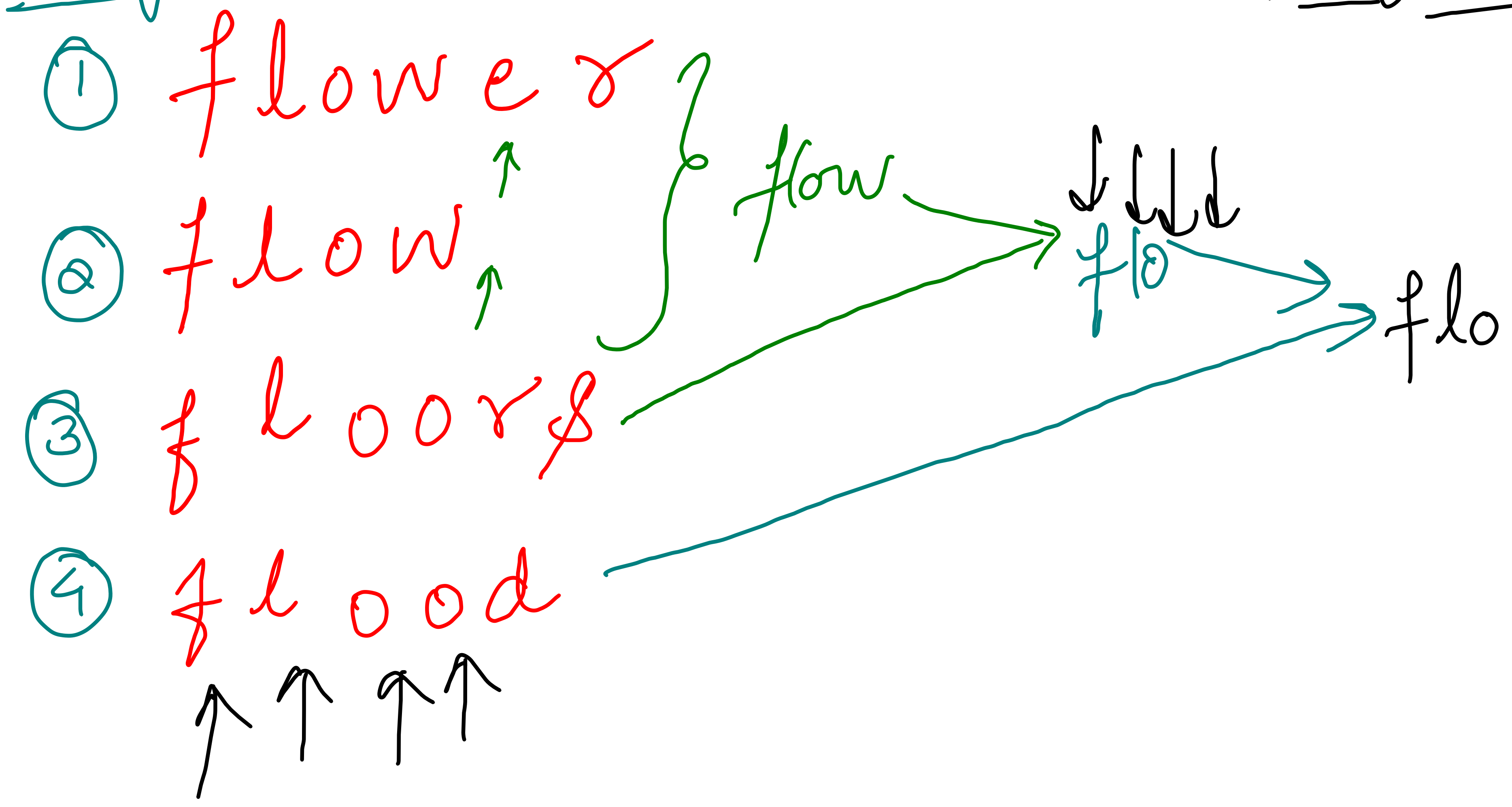
```java
public static int reverse(int x) {
  int rev = 0;
  while(x != 0){
      int dig = x % 10;
      int temp = rev * 10 + dig;

      if(rev != (temp - dig) / 10){
          return 0;
      }

      rev = temp;
      x = x/10;
  }

  return rev;
}
```

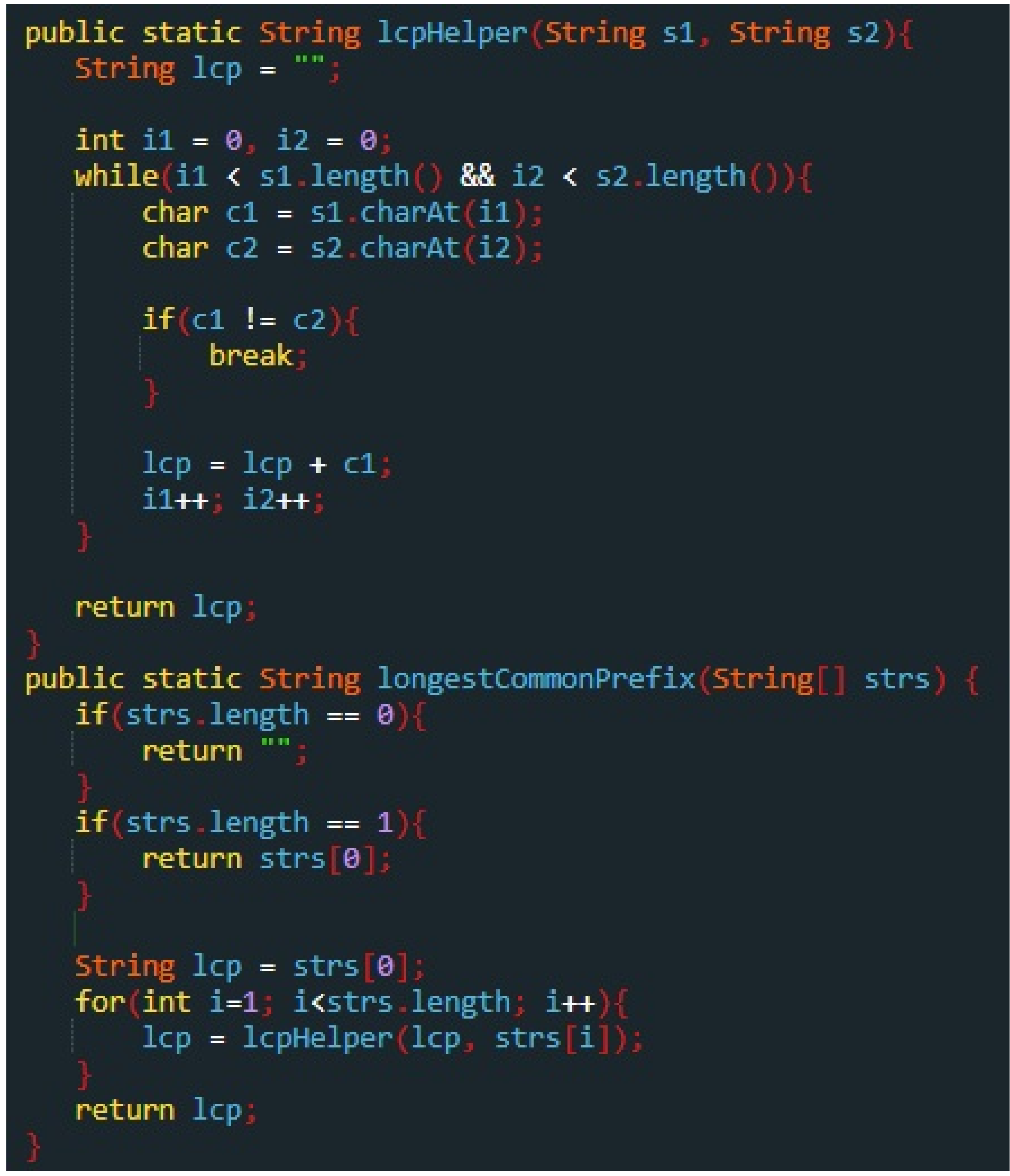
Longest Common Prefix

1. ✓ Horizontal Scanning
2. Vertical Scanning
3. Divide & Conquer
4. Queries using Trie

Strings

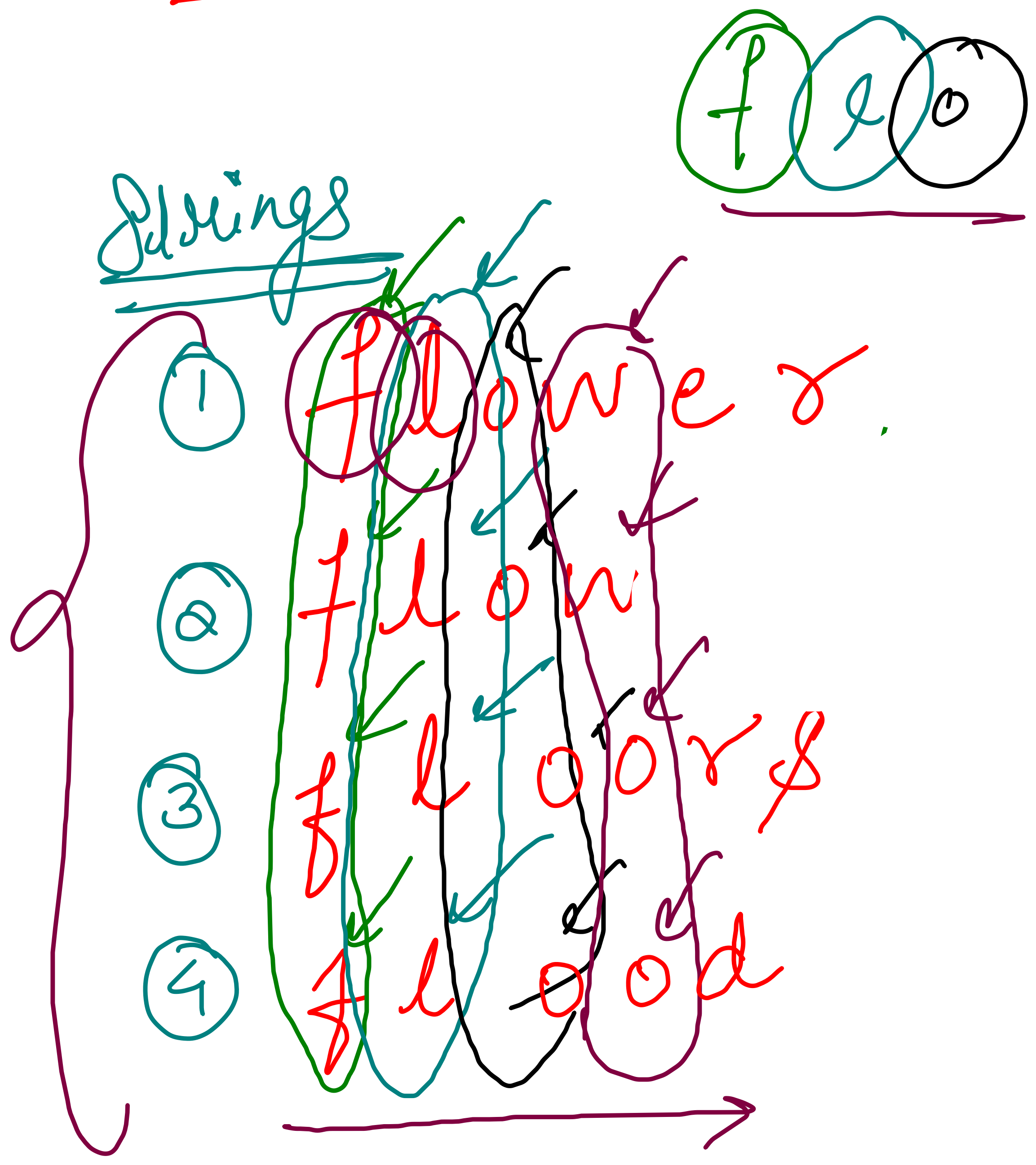
Horizontal Scanning



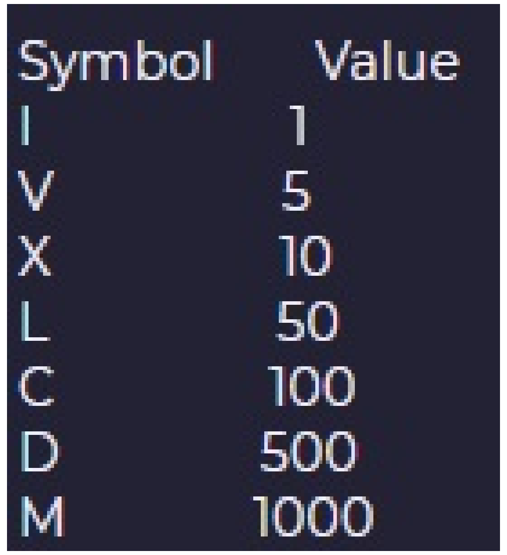


```java
public static String lcpHelper(String s1, String s2){
   String lcp = "";

   int i1 = 0, i2 = 0;
   while(i1 < s1.length() && i2 < s2.length()){
       char c1 = s1.charAt(i1);
       char c2 = s2.charAt(i2);

       if(c1 != c2){
           break;
       }

       lcp = lcp + c1;
       i1++; i2++;
   }

   return lcp;
}
public static String longestCommonPrefix(String[] strs) {
   if(strs.length == 0){
       return "";
   }
   if(strs.length == 1){
       return strs[0];
   }

   String lcp = strs[0];
   for(int i=1; i<strs.length; i++){
       lcp = lcpHelper(lcp, strs[i]);
   }
   return lcp;
}
```

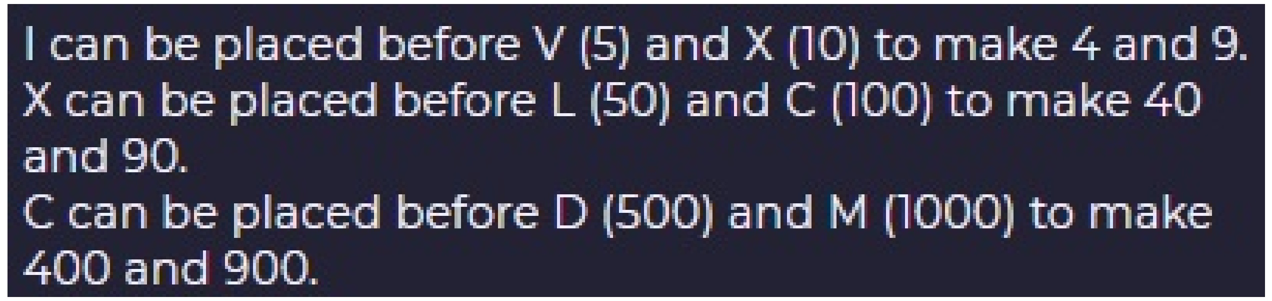
Vertical Scanning



Integer to Roman



Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000



I can be placed before V (5) and X (10) to make 4 and 9.
X can be placed before L (50) and C (100) to make 40 and 90.
C can be placed before D (500) and M (1000) to make 400 and 900.

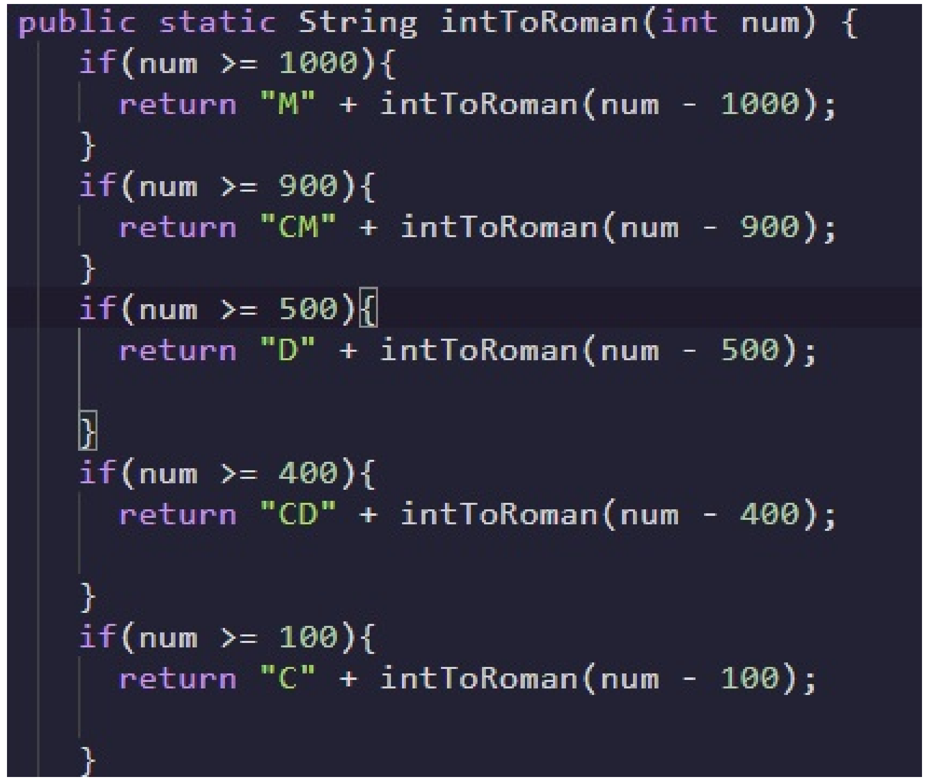
I,	IV,	V,	IX,	X,	XL,	L,	XC,	C,
1	4	5	9	10	40	50	90	100

CD,	D,	CM,	M
400	500	900	1000

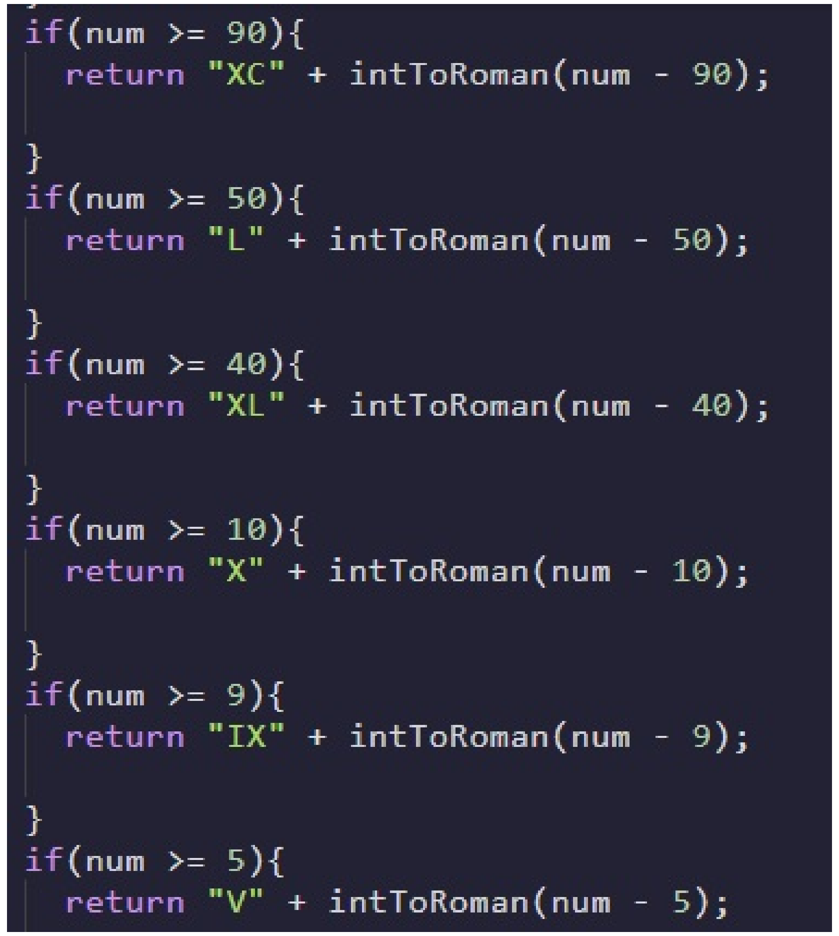
RfI(num)

$\hookrightarrow$ 'y' + RfnI(x)

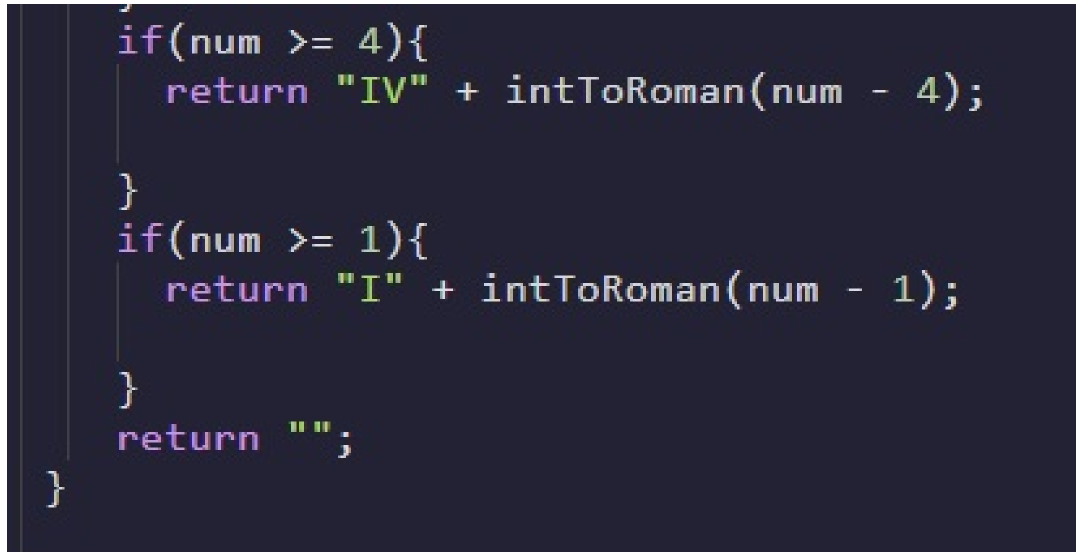
Solution



```java
public static String intToRoman(int num) {
    if(num >= 1000){
      return "M" + intToRoman(num - 1000);
    }
    if(num >= 900){
      return "CM" + intToRoman(num - 900);
    }
    if(num >= 500){
      return "D" + intToRoman(num - 500);

    }
    if(num >= 400){
      return "CD" + intToRoman(num - 400);

    }
    if(num >= 100){
      return "C" + intToRoman(num - 100);

    }
```



```java
if(num >= 90){
  return "XC" + intToRoman(num - 90);

}
if(num >= 50){
  return "L" + intToRoman(num - 50);

}
if(num >= 40){
  return "XL" + intToRoman(num - 40);

}
if(num >= 10){
  return "X" + intToRoman(num - 10);

}
if(num >= 9){
  return "IX" + intToRoman(num - 9);

}
if(num >= 5){
  return "V" + intToRoman(num - 5);
```

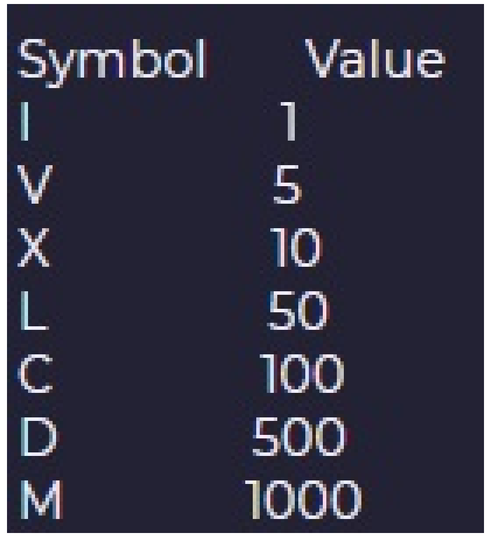


```java
    if(num >= 4){
      return "IV" + intToRoman(num - 4);

    }
    if(num >= 1){
      return "I" + intToRoman(num - 1);

    }
    return "";
}
```

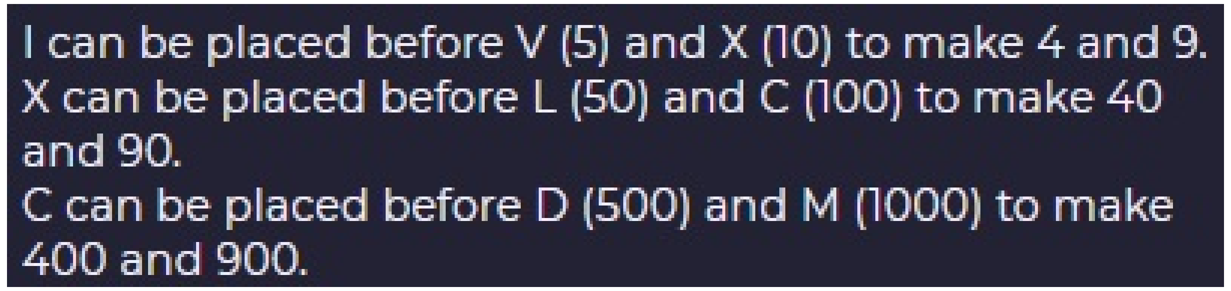
Roman to Integer

I,	IV,	V,	IX,	X,	XL,	L,	XC,	C,
1	4	5	9	10	40	50	90	100

CD,	D,	CM,	M
400	500	900	1000



Symbol	Value
I	1
V	5
X	10
L	50
C	100
D	500
M	1000

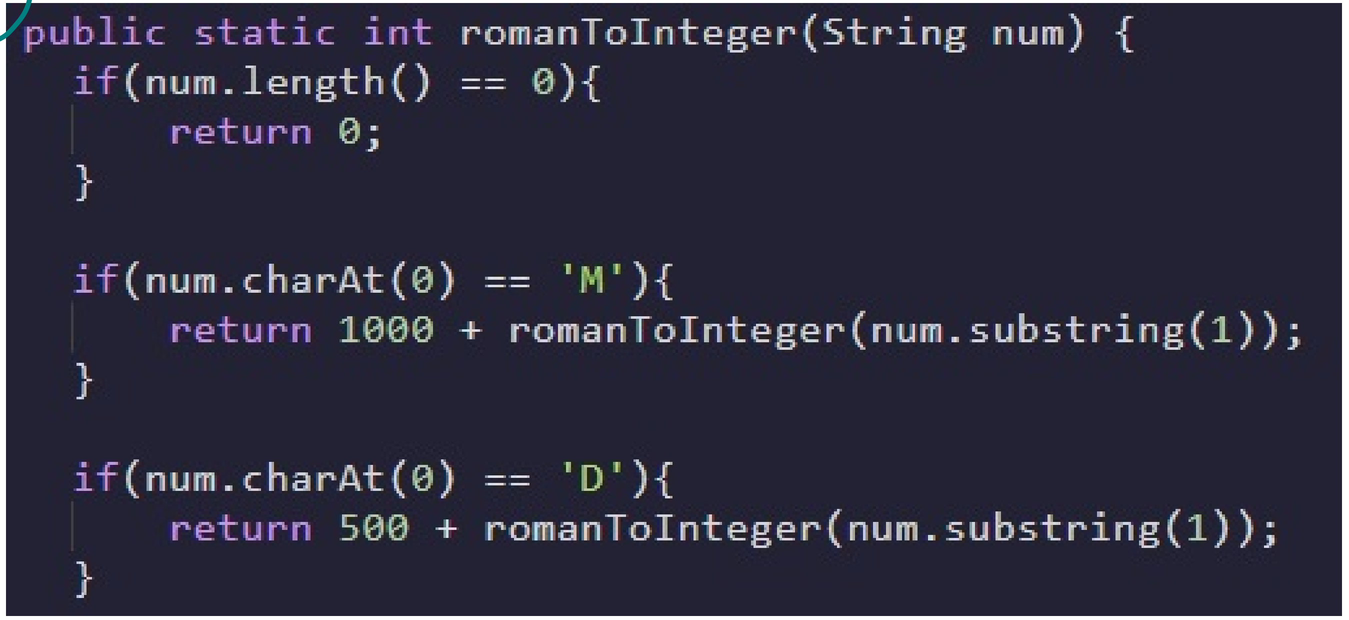


I can be placed before V (5) and X (10) to make 4 and 9.
X can be placed before L (50) and C (100) to make 40 and 90.
C can be placed before D (500) and M (1000) to make 400 and 900.

$\overset{\text{num}}{\overbrace{\text{int(char(s))}}}$ + RtoI(remaining substring)

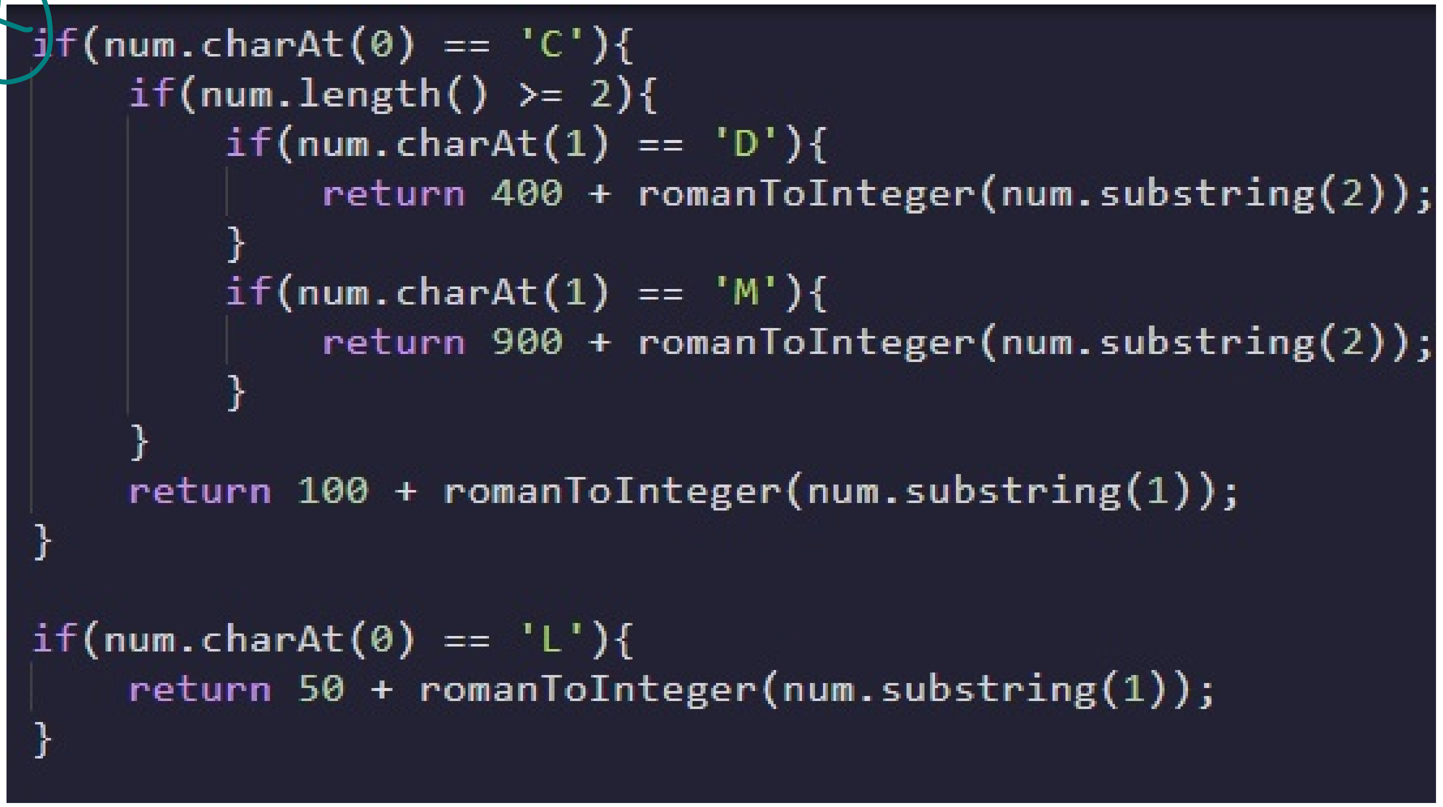
Code

1



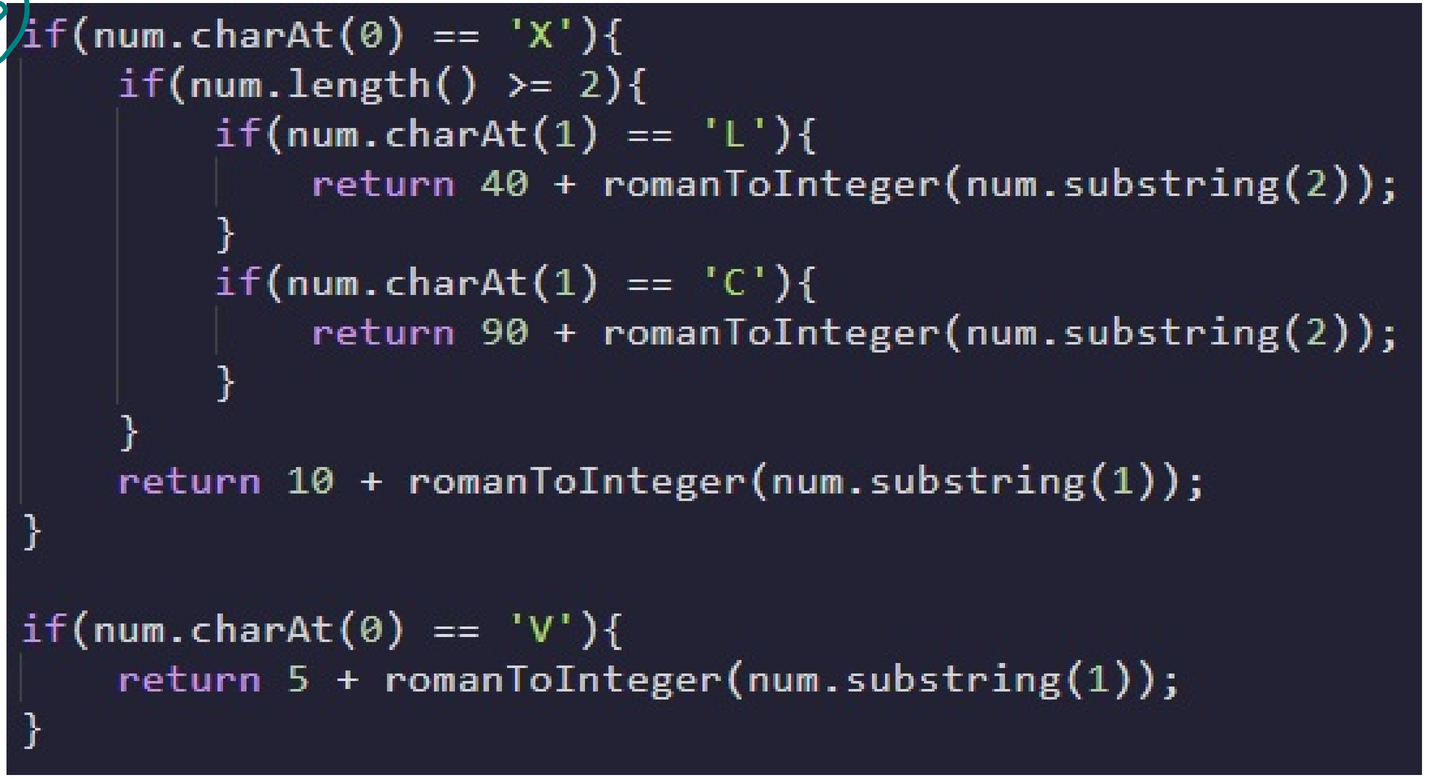
```java
public static int romanToInteger(String num) {
  if(num.length() == 0){
      return 0;
  }

  if(num.charAt(0) == 'M'){
      return 1000 + romanToInteger(num.substring(1));
  }

  if(num.charAt(0) == 'D'){
      return 500 + romanToInteger(num.substring(1));
  }
```

2



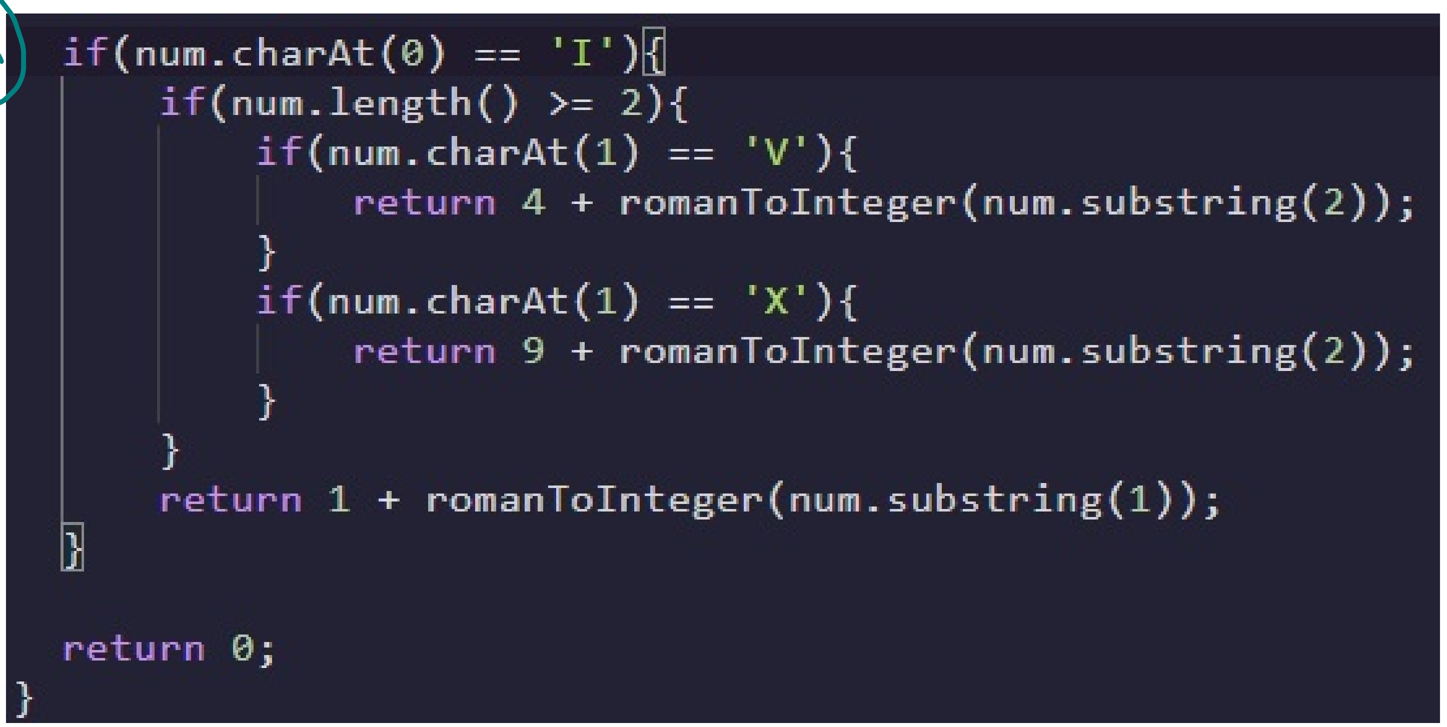
```java
if(num.charAt(0) == 'C'){
    if(num.length() >= 2){
        if(num.charAt(1) == 'D'){
            return 400 + romanToInteger(num.substring(2));
        }
        if(num.charAt(1) == 'M'){
            return 900 + romanToInteger(num.substring(2));
        }
    }
    return 100 + romanToInteger(num.substring(1));
}

if(num.charAt(0) == 'L'){
    return 50 + romanToInteger(num.substring(1));
}
```

3



```java
if(num.charAt(0) == 'X'){
    if(num.length() >= 2){
        if(num.charAt(1) == 'L'){
            return 40 + romanToInteger(num.substring(2));
        }
        if(num.charAt(1) == 'C'){
            return 90 + romanToInteger(num.substring(2));
        }
    }
    return 10 + romanToInteger(num.substring(1));
}

if(num.charAt(0) == 'V'){
    return 5 + romanToInteger(num.substring(1));
}
```

4



```java
  if(num.charAt(0) == 'I'){
      if(num.length() >= 2){
          if(num.charAt(1) == 'V'){
              return 4 + romanToInteger(num.substring(2));
          }
          if(num.charAt(1) == 'X'){
              return 9 + romanToInteger(num.substring(2));
          }
      }
      return 1 + romanToInteger(num.substring(1));
  }

  return 0;
}
```

Example

Q

$1000 +$ CMXCIV

$1000 + 900 +$ XCIV

$1000 + 900 + 90 +$ IV

$1000 + 900 + 90 + 4 = \boxed{1994}$

1000

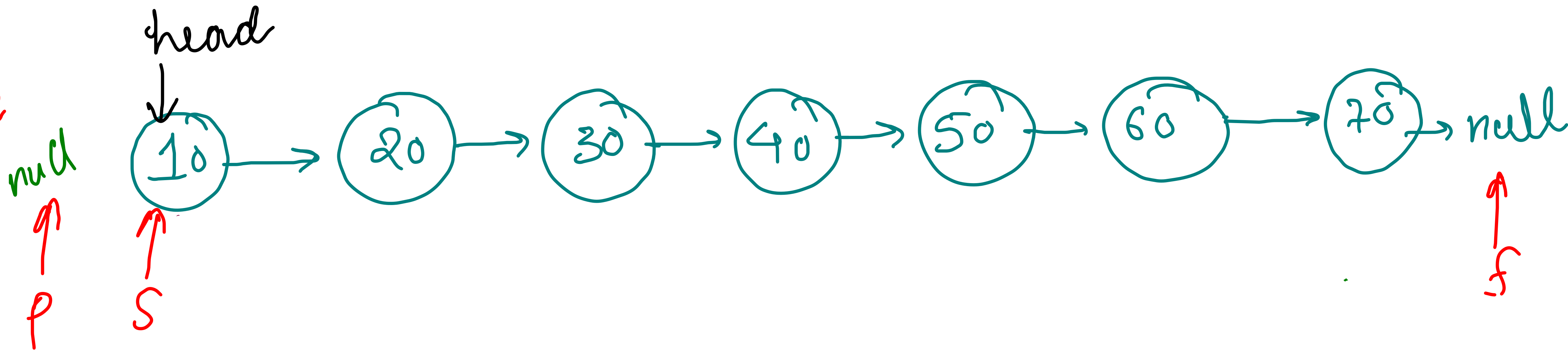
900

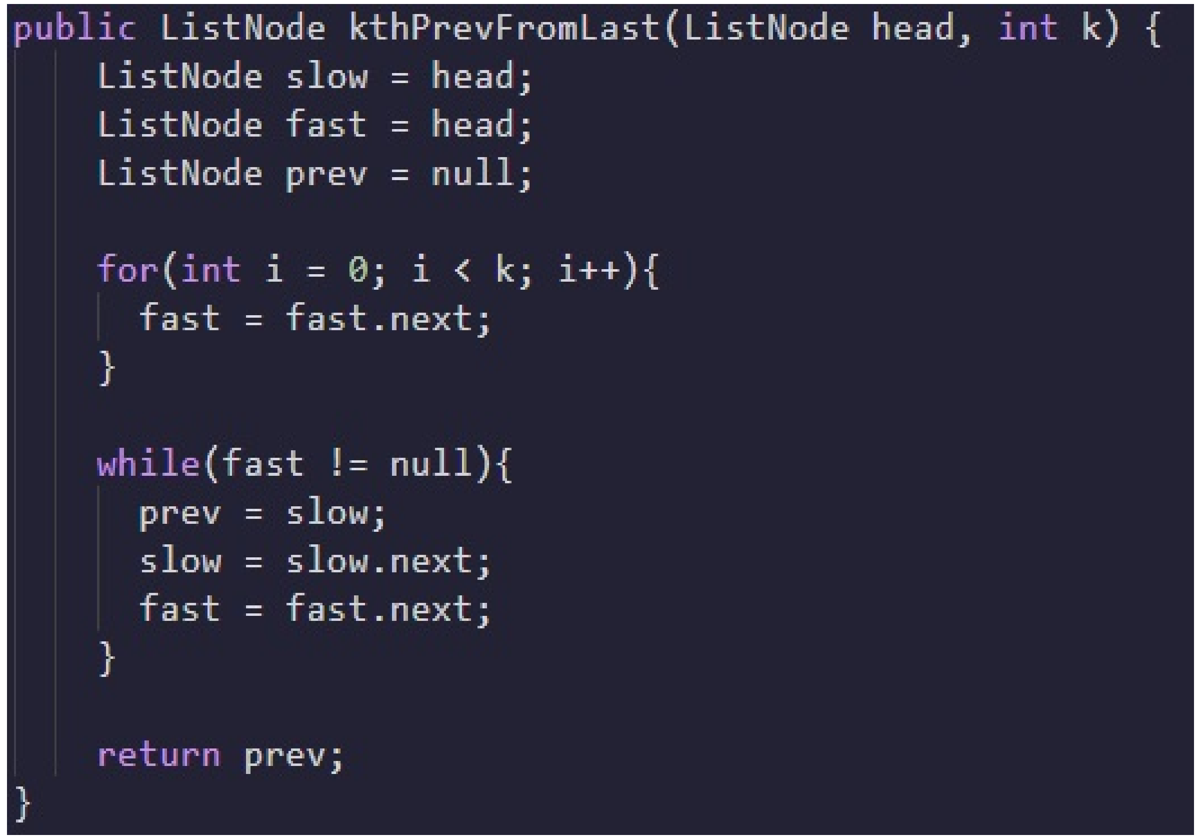
Remove Nth node from End of linked list

$1 \le N \le size$

$N = 3$

Getting kth node from end





```java
public ListNode kthPrevFromLast(ListNode head, int k) {
    ListNode slow = head;
    ListNode fast = head;
    ListNode prev = null;

    for(int i = 0; i < k; i++){
      fast = fast.next;
    }

    while(fast != null){
      prev = slow;
      slow = slow.next;
      fast = fast.next;
    }

    return prev;
}
```

(1) $k = 1$

(2) $k = 7$