|  |  |
| --- | --- |
| **Symbol** | **Value** |
| [I](http://en.wikipedia.org/wiki/I) | [1](http://en.wikipedia.org/wiki/1_(number)) |
| [V](http://en.wikipedia.org/wiki/V) | [5](http://en.wikipedia.org/wiki/5_(number)) |
| [X](http://en.wikipedia.org/wiki/X) | [10](http://en.wikipedia.org/wiki/10_(number)) |
| [L](http://en.wikipedia.org/wiki/L) | [50](http://en.wikipedia.org/wiki/50_(number)) |
| [C](http://en.wikipedia.org/wiki/C) | [100](http://en.wikipedia.org/wiki/100_(number)) |
| [D](http://en.wikipedia.org/wiki/D) | [500](http://en.wikipedia.org/wiki/500_(number)) |
| [M](http://en.wikipedia.org/wiki/M) | [1,000](http://en.wikipedia.org/wiki/1000_(number)) |

# Rules

**Valid Characters**

Only I, V, X, L, C, D, and M are valid characters.

**Repeating Characters**

M can be appear an infinite number of times

I, X, and C can appear up to 3 times.

V, L, and D can appear only 1 time.

**Order of Characters**

From left to right, characters should appear in descending order.

The following cases are exceptions.

1. A single I can come before V and X
2. A single X can come before L and C
3. A single C can come before D and M

The cases above can only appear as the last 2 characters in a roman numeral.

# Validation

The process will run a series of validations based on the rules above.

**Invalid Characters**

Returns false if a sequence of characters contains a character other than I, V, X, L, C, D, or M

//validateCharacters(char[] digits)

//Set validChars to string “IVXLCDM”

//Loop index from 0 to length of digits

//If validChars does not contain digits at index

//Return false

//Return true

**Character Counts**

Returns false if a character occurs an invalid number of times in a sequence

//validateCounts(char[] digits)

//Initialize a count for each possible character to 0

//Loop through each character in digits

//Increment count for current character

//If count for ‘I’, ‘X’, or ‘C’ exceeds 3

//Return false

//If count for ‘V’, ‘L’, or ‘D’ exceeds 1

//Return false

//Return true

**Decending Order**

Returns false if characters do not appear in descending order. Allows for subtraction ordering, such as ‘IV’, or ‘XL’.

//validateOrder(char[] digits)

//Set lastChar to enumeration of M

//Loop index from 0 to length of digits

//Set current to enumeration of digits at index

// If current > lastChar

//If current can not be subtracted by lastChar

//Return false

//If current is not the last digit

//Return false

//Set lastChar equal to current

//Return true

# Conversion

**String to int**

Converts a string to an int. Traverses string from right to left, keeping track of the highest digit encountered. If a digit is greater than or equal to the highest digit encountered, its value is added to the sum. If a digit is less than the highest digit encountered, it’s value is subtracted from the sum.