Removing Characters from String

We cannot remove characters from the string. We need to create a new string skipping the characters that are to be removed.

Example 1: Removing first three characters from string

```
1  String str = "trisect";
2  int len = str.length();
3  String newStr = "";                  //empty string
4
5  for(int i = 3; i < len; i++)         //string loop from 4th position (index = 3)
6  {
7     char ch = str.charAt(i);
8     newStr = newStr + ch;             //adding characters to new string
9  }
10 System.out.println(newStr);
```

Output:

```
sect
```

Before loop	str = "trisect" len = 7 newStr = ""		
loop variable	**ch**	**newStr**	**Print**
i = 3	's'	⇨ newStr = "" + 's' ⇨ newStr = "s"	-
i = 4	'e'	⇨ newStr = "s" + 'e' ⇨ newStr = "se"	-
i = 5	'c'	⇨ newStr = "se" + 'c' ⇨ newStr = "sec"	-
i = 6	't'	⇨ newStr = "sec" + 't' ⇨ newStr = "sect"	-
i = 7	-	-	-
Outside Loop	-	-	sect

Problem 1	**Give output of following code.**

```
1  String str = "world";
2  String qstr = "##"
3  int k = 2;
4  int p = str.length();
5
6  for(int i = k; i < p; i++)
```

```
7  {
8      char rch = str.charAt(i);
9      qstr = qstr + ':' + rch;
10 }
11 System.out.println(qstr);
```

Not Equal Operator (!=)

Example 2: Consider a very simple problem statement:
`Check if a number is not divisible by 3.`

We can solve the problem statement above in using not equal operator:

1. Check number is not a multiple of 3 => `Condition: (num%3 != 0)`

!= gives true when the LHS and RHS are not same.

Problem 2	**Give output of condition given below based on various values following them.**
Condition	`(x != 10)`
Input	x = 10
	x = 15

Example 3: Removing character 'n' from string

```
1  String str = "banana";
2  int len = str.length();
3  String newStr = "";                  //empty string
4
5  for(int i = 0; i < len; i++)
6  {
7     char ch = str.charAt(i);
8     if(ch != 'n')                     //only when character is not equal to 'n'
9     {
10          newStr = newStr + ch;       //adding characters except 'n' to new string
11    }
12 }
13 System.out.println(newStr);
```

Output:

baaa

Before loop	str = "banana" len = 6 newStr = ""		
loop variable	**ch**	**If**	**Print**
i = 0	'b'	⇨ 'b' != 'n' => true ⇨ newStr = "" + 'b' ⇨ newStr = "b"	-

i = 1	'a'	⇨ 'a' != 'n' => true ⇨ newStr = "b" + 'a' ⇨ newStr = "ba"	-
i = 2	'n'	⇨ 'n' != 'n' => false	-
i = 3	'a'	⇨ 'a' != 'n' => true ⇨ newStr = "ba" + 'a' ⇨ newStr = "baa"	-
i = 4	'n'	⇨ 'n' != 'n' => false	-
i = 5	'a'	⇨ 'a' != 'n' => true ⇨ newStr = "baa" + 'a' ⇨ newStr = "baaa"	-
i = 6	-	-	-
Outside Loop	-	-	baaa

Problem 3 | **Give output of following code.**

```
1  String str = "kicked";
2  int len = str.length();
3  int x = 0;
4  for(int i = 0; i < len; i++)
5  {
6      char ch = str.charAt(i);
7      if(ch != 'k')
8      {
9              x = x + 1;
10     }
11 }
12 System.out.println(x);
```

Logical Operators

AND (&&)

Example 4: Consider a very simple problem statement:
`Check and print "Hurray" if a number is less than 20 and odd.`

We can split the problem statement above into following steps:

1. Check number is less than 20 => `Condition 1: (num < 20)`
2. Check number is odd => `Condition 2: (num%2 != 0)`
3. Print "Hurray" if both condition 1 **and** condition 2 are satisfied.

In situations like these where we want to do something when both the conditions evaluate to true AND operator (&&) can be used to join these two conditions. So the condition 1 and 2 for above problem statement can be joined using && like follows:

```
Condition 3: (num < 20) && (num % 2 != 0)

if num = 31,        condition 3: false        because      num < 20     => false
if num = 14,        condition 3: false        because      num%2 != 0   => false
```

```
if num = 47,        condition 3: false        because      both are false
if num = 15,        condition 3: true         because      both are true
```

&& gives true only if both sides of it are true. Otherwise it gives false.

Truth Table AND (&&)		
Condition 1	**Condition 2**	**Result**
true	false	false
false	true	false
false	false	false
true	true	true

Example 5: Write condition for following statement

```
number is between 20 and 50 (inclusive)
```

We can split the problem statement above into following 2 conditions:

1. Check number is greater than or equal to 20 => `Condition 1: (num >= 20)`
2. Check number is less than or equal to 50 => `Condition 2: (num <= 50)`

We have to check for both so final condition:

```
Condition 3: num >= 20 && num <= 50
```

```
if num = 70,        condition 3: false        because      num <= 50    => false
if num = 10,        condition 3: false        because      num >= 20    => false
if num = 23,        condition 3: true         because      both are true
```

Problem 4.1	**Give output of condition given below based on various values following them.**
Condition	`(x > 0) && (x > y)`
Input	x = 10, y = 5
	x = 0, y = -1
	x = 10, y = 20
Problem 4.2	**Give output of condition given below based on various values following them.**
Condition	`(x > y) && (x + y > 100)`
Input	x = 20, y = 30
	x = 50, y = 60
	x = 50, y = 50
	x = 75, y = 30

Quick Sheet: Working with Characters

Conditions	Code

For selecting Uppercase characters ['A' – 'Z']	`ch >= 'A' && ch <= 'Z'`
	
	`will give true only when ch comes after 'A' and before 'Z' (between 'A' and 'Z')`

For selecting Lowercase characters ['a' – 'z']	`ch >= 'a' && ch <= 'z'`

	will give true only when ch comes between 'a' and 'z'
For selecting Digits ['0' – '9']	ch >= '0' && ch <= '9' will give true only when ch comes between '0' and '9'

Example 6: Replace all characters that are between character 'A' and 'Z' (uppercase characters) with '#'.

```
1  String str = "StaR";
2  int len = str.length();
3  String newStr = "";
4  for(int i = 0; i < len; i++)
5  {
6      char ch = str.charAt(i);
7      if(ch >= 'A' && ch <= 'Z')              //checking uppercase character
8      {
9              newStr = newStr + '#';
10     }
11     else
12     {
13             newStr = newStr + ch;
14     }
15 }
16 System.out.println(newStr);
```

Output:

```
#ta#
```

Dry Run:

Before loop	str = "StaR" len = 4 newStr = ""			
loop variable	**ch**	**If**	**else**	**Print**
i = 0	'S'	⇨ true ⇨ newStr = "" + '#' ⇨ newStr = "#"	-	-
i = 1	't'	⇨ false	⇨ newStr = "#" + 't' ⇨ newStr = "#t"	-
i = 2	'a'	⇨ false	⇨ newStr = "#t" + 'a' ⇨ newStr = "#ta"	-
i = 3	'R'	⇨ true ⇨ newStr = "#ta" + '#' ⇨ newStr = "#ta#"	-	-
i = 4	-	-	-	-
Outside Loop	-	-	-	#ta#

Problem 5	Give output of following code.

```
1  String str = "trisectinstitute";
2  String ystr = "";
3  int len = str.length();
4
5  for(int i = 0; i < len; i++)
6  {
7      char ch1 = str.charAt(i);
8      if(i%2 == 0 && i%3 == 0)
9      {
10           ystr = ystr + "$";
11     }
12     ystr = ystr + ch1;
13 }
14 System.out.println(ystr);
```

OR (||)

Example 7: Consider another simple problem statement:
`Check and print "Yay" if a number is greater than 10 or even.`

We can split the problem statement above into following steps:
1. Check number is greater than 10 => `Condition 1: (num > 10)`
2. Check number is even => `Condition 2: (num%2 == 0)`
3. Print "Yay" if either condition 1 **or** condition 2 is satisfied.

In situations like these where we want to do something when any of the conditions evaluate to true OR operator (||) can be used to join these two conditions. So the condition 1 and 2 for above problem statement can be joined using || like follows:

```
Condition 3: (num > 10) || (num % 2 == 0)
```

```
if num = 16,        condition 3: true        because      both are true
if num = 13,        condition 3: true        because      num > 10     => true
if num = 4,         condition 3: true        because      num%2 == 0   => true
if num = 5,         condition 3: false       because      both are false
```

|| gives true when any of its side is true. Otherwise it gives false only when both the sides are false.

Truth Table OR (||)		
Condition 1	**Condition 2**	**Result**
true	false	true
false	true	true
true	true	true
false	false	false

Problem 6	Give output of condition given below based on various values following them.
Condition	`(x > y) || (z > x)`
Input	x = 10, y = 15, z = 5
	x = 13, y = 15, z = 17
	x = 10, y = 9, z = 8

Example 8: Replace all characters that are either 'a' or 'b' with 'x'.

```
1  String str = "bad";
2  int len = str.length();
3  String newStr = "";
4  for(int i = 0; i < len; i++)
5  {
6     char ch = str.charAt(i);
7     if(ch == 'a' || ch =='b')
8     {
9           newStr = newStr + 'x';
10    }
11    else
12    {
13          newStr = newStr + ch;
14    }
15 }
16 System.out.println(newStr);
```

Output:

xxd

Dry Run:

Before loop	str = "bad" len = 3 newStr = ""			
loop variable	**ch**	**if**	**else**	**Print**
i = 0	'b'	⇨ true ⇨ newStr = "" + 'x' ⇨ newStr = "x"	-	-
i = 1	'a'	⇨ true ⇨ newStr = "x" + 'x' ⇨ newStr = "xx"	-	-
i = 2	'd'	false	⇨ newStr = "xx" + 'd' ⇨ newStr = "xxd"	-
i = 3	-	-	-	-
Outside Loop	-	-	-	xxd

Problem 7 **Give output of following code.**

```
1  String str = "something";
2  String xstr = "";
3  int len = str.length();
4
5  for(int i = 0; i < len; i++)
6  {
7     char ch1 = str.charAt(i);
8     if(i < 3 || i%3 == 0)
9     {
10          xstr = xstr + "@";
11    }
12    else
13    {
14          xstr = xstr + ch1;
15    }
16 }
17 System.out.println(xstr);
```