STRING - I

1. Given a string name, e.g. "Bob", return a greeting of the form "Hello Bob!".   
     
   helloName("Bob") → "Hello Bob!"  
   helloName("Alice") → "Hello Alice!"  
   helloName("X") → "Hello X!"

Solution:

public String helloName(String name) {

return "Hello " + name + "!";

}

1. Given an "out" string length 4, such as "<<>>", and a word, return a new string where the word is in the middle of the out string, e.g. "<<word>>". Note: use str.substring(i, j) to extract the String starting at index i and going up to but not including index j.

Solution:

public String makeOutWord(String out, String word) {

return(out.substring(0,2) + word + out.substring(2,4));

}

1. The web is built with HTML strings like "<i>Yay</i>" which draws Yay as italic text. In this example, the "i" tag makes <i> and </i> which surround the word "Yay". Given tag and word strings, create the HTML string with tags around the word, e.g. "<i>Yay</i>".

Solution:

public String makeTags(String tag, String word) {

return ('<' + tag + '>' + word + '<' + '/' + tag + '>');

}

1. Given 2 strings, a and b, return a new string made of the first char of a and the last char of b, so "yo" and "java" yields "ya". If either string is length 0, use '@' for its missing char.   
     
   lastChars("last", "chars") → "ls"  
   lastChars("yo", "java") → "ya"  
   lastChars("hi", "") → "h@"

Solution:

public String lastChars(String a, String b) {

String amp = "@";

if (a.length()== 0 && b.length() > 0)

return (amp + b.substring(b.length()-1));

else if (a.length() > 0 && b.length() == 0)

return ((a.substring(0,1)) + amp);

else if (a.length() > 0 && b.length() > 0)

return (a.substring(0,1) + b.substring(b.length()-1));

return amp + amp;

}

OR

public String lastChars(String a, String b) {

int lenB = b.length();

String combStr = "";

combStr += (a.length() >= 1) ? a.charAt(0) : '@';

combStr += (lenB >= 1) ? b.charAt(lenB-1) : '@';

return combStr;

}

1. Given a string, return a version without the first and last char, so "Hello" yields "ell". The string length will be at least 2.   
     
   withoutEnd("Hello") → "ell"  
   withoutEnd("java") → "av"  
   withoutEnd("coding") → "odin"

Solution:

public String withoutEnd(String str) {

//if (str.length() >= 1)

return (str.substring(1,str.length()-1));

//else

// return "";

}

1. Given a string, return true if "bad" appears starting at index 0 or 1 in the string, such as with "badxxx" or "xbadxx" but not "xxbadxx". The string may be any length, including 0. Note: use .equals() to compare 2 strings.   
     
   hasBad("badxx") → true  
   hasBad("xbadxx") → true  
   hasBad("xxbadxx") → false

Solution:

public boolean hasBad(String str)

{

int len = str.length();

if(len == 3 && str.equals("bad"))

return true;

else if(len >= 4)

{

if(str.substring(0, 3).equals("bad"))

return true;

return str.substring(1, 4).equals("bad");

}

else

return false;

}

OR

public boolean hasBad(String str) {

if (str.length()>=3 && str.substring(0, 3).equals("bad")) {

return true;

}

if (str.length()>=4 && str.substring(1, 4).equals("bad")) {

return true;

}

return false;

}

1. Given a string, if the first or last chars are 'x', return the string without those 'x' chars, and otherwise return the string unchanged.   
     
   withoutX("xHix") → "Hi"  
   withoutX("xHi") → "Hi"  
   withoutX("Hxix") → "Hxi"

Solution:

public String withoutX(String str) {

if (str.length() > 0 && str.charAt(0) == 'x') {

str = str.substring(1);

}

if (str.length() > 0 && str.charAt(str.length()-1) == 'x') {

str = str.substring(0, str.length()-1);

}

return str;

}

OR

public String withoutX(String str) {

int len = str.length();

if(len >= 2)

{

char ch = str.charAt(0);

StringBuilder stbuild = new StringBuilder(len);

if(ch != 'x')

stbuild.append(ch);

stbuild.append(str.substring(1, len-1));

ch = str.charAt(len-1);

if(ch != 'x')

stbuild.append(ch);

return stbuild.toString();

}

else if(len == 1 && str.charAt(0) == 'x')

return "";

else

return str;

}

1. Given a string, return the string made of its first two chars, so the String "Hello" yields "He". If the string is shorter than length 2, return whatever there is, so "X" yields "X", and the empty string "" yields the empty string "". Note that str.length() returns the length of a string.   
     
   firstTwo("Hello") → "He"  
   firstTwo("abcdefg") → "ab"  
   firstTwo("ab") → "ab"

Solution:

public String firstTwo(String str) {

if (str.length() >= 2)

return (str.substring(0,2));

else if (str.length() == 1)

return str;

else

return "";

}

OR

public String firstTwo(String str) {

if (str.length() >= 2) {

return str.substring(0, 2);

} else {

return str;

}

}

1. Given two strings, a and b, return the result of putting them together in the order abba, e.g. "Hi" and "Bye" returns "HiByeByeHi".   
     
   makeAbba("Hi", "Bye") → "HiByeByeHi"  
   makeAbba("Yo", "Alice") → "YoAliceAliceYo"  
   makeAbba("What", "Up") → "WhatUpUpWhat"

Solution:

public String makeAbba(String a, String b) {

return a + b + b + a;

}

1. Given a string, return a string length 2 made of its first 2 chars. If the string length is less than 2, use '@' for the missing chars.   
     
   atFirst("hello") → "he"  
   atFirst("hi") → "hi"  
   atFirst("h") → "h@"

Solution:

public String atFirst(String str) {

if (str.length() >= 2)

return (str.substring(0,2));

else if (str.length() == 1)

return str + "@";

else

return "@" + "@";

}

1. Given a string of odd length, return the string length 3 from its middle, so "Candy" yields "and". The string length will be at least 3.   
     
   middleThree("Candy") → "and"  
   middleThree("and") → "and"  
   middleThree("solving") → "lvi"

Solution:

public String middleThree(String str) {

int len = str.length()/2;

return str.substring(len-1,len+2);

}

1. Given a string of even length, return the first half. So the string "WooHoo" yields "Woo".   
     
   firstHalf("WooHoo") → "Woo"  
   firstHalf("HelloThere") → "Hello"  
   firstHalf("abcdef") → "abc"

Solution:

public String firstHalf(String str) {

String result = "";

String half = str.substring(0, str.length()/2);

return result + half;

}

1. Given 2 strings, a and b, return a string of the form short+long+short, with the shorter string on the outside and the longer string on the inside. The strings will not be the same length, but they may be empty (length 0).   
     
   comboString("Hello", "hi") → "hiHellohi"  
   comboString("hi", "Hello") → "hiHellohi"  
   comboString("aaa", "b") → "baaab"

Solution:

public String comboString(String a, String b) {

if (a.length() > b.length())

return b + a + b;

else

return a + b + a;

}

1. Given a string, return true if it ends in "ly".   
     
   endsLy("oddly") → true  
   endsLy("y") → false  
   endsLy("oddy") → false

Solution:

public boolean endsLy(String str) {

if (str.length() >= 2)

return (str.substring(str.length()- 2).equals("ly"));

else

return false;

}

1. Given a string, return a new string made of 3 copies of the first 2 chars of the original string. The string may be any length. If there are fewer than 2 chars, use whatever is there.   
     
   extraFront("Hello") → "HeHeHe"  
   extraFront("ab") → "ababab"  
   extraFront("H") → "HHH"

Solution:

public String extraFront(String str) {

if (str.length()>= 3)

//String result = str.substring(0, 2);

return str.substring(0, 2)+ str.substring(0, 2)+ str.substring(0, 2);

else

return str + str + str;

}

OR

|  |
| --- |
| public String extraFront(String str) |
|  |

|  |
| --- |
|  |
| { |
|  |
|  |
| String temp; |
|  |
|  |
| if(str.length() >= 3) |
|  |
|  |
| temp = str.substring(0, 2); |
|  |
|  |
| else |
|  |
|  |
| temp = str; |
|  |
|  |
| return (temp + temp + temp); |
|  |

|  |
| --- |
| } |

1. Given a string, return a "rotated left 2" version where the first 2 chars are moved to the end. The string length will be at least 2.   
     
   left2("Hello") → "lloHe"  
   left2("java") → "vaja"  
   left2("Hi") → "Hi"

Solution:

public String left2(String str) {

String temp;

if (str.length() > 2)

temp = str.substring(2) + str.substring(0,2);

else

temp = str;

return temp;

}

OR

public String left2(String str)

{ return (str.substring(2) + str.substring(0, 2));

}

1. Given a string of even length, return a string made of the middle two chars, so the string "string" yields "ri". The string length will be at least 2.   
     
   middleTwo("string") → "ri"  
   middleTwo("code") → "od"  
   middleTwo("Practice") → "ct"

Solution:

public String middleTwo(String str) {

int len = str.length()/2;

String mid = str.substring(len - 1 , len + 1);

return mid;

}

1. Given 2 strings, return their concatenation, except omit the first char of each. The strings will be at least length 1.   
     
   nonStart("Hello", "There") → "ellohere"  
   nonStart("java", "code") → "avaode"  
   nonStart("shotl", "java") → "hotlava"

Solution:

public String nonStart(String a, String b) {

return (a.substring(1) + b.substring(1));

}

1. Given a string and a second "word" string, we'll say that the word matches the string if it appears at the front of the string, except its first char does not need to match exactly. On a match, return the front of the string, or otherwise return the empty string. So, so with the string "hippo" the word "hi" returns "hi" and "xip" returns "hip". The word will be at least length 1.   
     
   startWord("hippo", "hi") → "hi"  
   startWord("hippo", "xip") → "hip"  
   startWord("hippo", "i") → "h"

Solution:

public String startWord(String str, String word) {

int lenStr = str.length();

int lenWord = word.length();

String temp;

if(lenStr >= lenWord){

temp = str.substring(1, lenWord);

if (word.substring(1).equals(temp))

return str.charAt(0) + temp;

else

return "";

}

else

return "";

}

1. Given a string, return true if the first 2 chars in the string also appear at the end of the string, such as with "edited".   
     
   frontAgain("edited") → true  
   frontAgain("edit") → false  
   frontAgain("ed") → true

Solution:

public boolean frontAgain(String str) {

String temp;

int len = str.length();

if (len >= 2){

temp = str.substring(len - 2, len);

return(str.substring(0,2).equals(temp));

}else

return false;

}

OR

public boolean frontAgain(String str) {

if (str.length() < 2) {

return false; // screen out the short-length case

}

String front = str.substring(0, 2);

String back = str.substring(str.length()-2);

return(front.equals(back));

// Solution notes: First screen out the length < 2 case to avoid index

// problems with the later code.

// Grab the two strings with substring() and compare them

// with equals(). Return the result of equals() directly -- don't need

// check if it's true or false. Here the two strings are stored in local

// variables which makes the code simpler but longer.

}

1. Given a string, return a version without both the first and last char of the string. The string may be any length, including 0.   
     
   withouEnd2("Hello") → "ell"  
   withouEnd2("abc") → "b"  
   withouEnd2("ab") → ""

Solution:

public String withouEnd2(String str) {

if (str.length() >= 3)

return(str.substring(1,str.length()-1));

else

return "";

}

1. Given a string, return a version without the first 2 chars. Except keep the first char if it is 'a' and keep the second char if it is 'b'. The string may be any length. Harder than it looks.   
     
   deFront("Hello") → "llo"  
   deFront("java") → "va"  
   deFront("away") → "aay"

Solution:

public String deFront(String str)

{

int len = str.length();

if(len >= 2)

{

StringBuilder stbuild = new StringBuilder(len);

if(str.charAt(0) == 'a')

stbuild.append('a');

if(str.charAt(1) == 'b')

stbuild.append('b');

stbuild.append(str.substring(2));

return stbuild.toString();

}

else if(len == 1 && str.charAt(0) == 'a')

return "a";

else

return "";

}

OR

public String deFront(String str) {

int len = str.length();

String temp = "";

for (int i = 0; i < len; i++) {

if (i == 0 && str.charAt(i) == 'a')

temp += 'a';

else if (i == 1 && str.charAt(i) == 'b')

temp += 'b';

else if (i > 1)

temp += str.charAt(i);

}

return temp;

}

1. Given a string, return a "rotated right 2" version where the last 2 chars are moved to the start. The string length will be at least 2.   
     
   right2("Hello") → "loHel"  
   right2("java") → "vaja"  
   right2("Hi") → "Hi"

Solution:

public String right2(String str) {

int len = str.length();

String result = "";

if (len >= 2)

result = str.substring( len - 2) + str.substring(0, len -2) ;

return result;

}

1. Given two strings, append them together (known as "concatenation") and return the result. However, if the concatenation creates a double-char, then omit one of the chars, so "abc" and "cat" yields "abcat".   
     
   conCat("abc", "cat") → "abcat"  
   conCat("dog", "cat") → "dogcat"  
   conCat("abc", "") → "abc"

Solution:

public String conCat(String a, String b) {

int aLen = a.length();

int bLen = b.length();

if(aLen>= 1 && bLen>= 1)

{

if(a.charAt(aLen-1) == b.charAt(0))

return (a + b.substring(1));

else

return (a + b);

}

return (a + b);

}

1. Given a string, if one or both of the first 2 chars is 'x', return the string without those 'x' chars, and otherwise return the string unchanged. This is a little harder than it looks.   
     
   withoutX2("xHi") → "Hi"  
   withoutX2("Hxi") → "Hi"  
   withoutX2("Hi") → "Hi"

Solution:

public String withoutX2(String str) {

String temp = "";

for (int i = 0; i < str.length(); i++) {

if (i == 0 && str.charAt(i) != 'x')

temp += str.charAt(i);

else if (i == 1 && str.charAt(i) != 'x')

temp += str.charAt(i);

else if (i > 1)

temp += str.charAt(i);

}

return temp;

}

1. Given a string, return a string length 1 from its front, unless **front** is false, in which case return a string length 1 from its back. The string will be non-empty.   
     
   theEnd("Hello", true) → "H"  
   theEnd("Hello", false) → "o"  
   theEnd("oh", true) → "o"

Solution:

public String theEnd(String str, boolean front) {

if (front){

return str.substring(0,1);

}

else {

return str.substring(str.length()-1);

}

}

1. Given a string, if the string begins with "red" or "blue" return that color string, otherwise return the empty string.   
     
   seeColor("redxx") → "red"  
   seeColor("xxred") → ""  
   seeColor("blueTimes") → "blue"

Solution:

public String seeColor(String str) {

if (str.length() >= 4){

if (str.substring(0,4).equals("blue"))

return "blue";

else if (str.substring(0,3).equals("red"))

return "red";

else

return "";

} else if (str.length()== 3){

if (str.substring(0,3).equals("red"))

return "red";

else

return "";

}

else

return "";

}

OR

public String seeColor(String str) {

if (str.startsWith("red"))

return "red";

if (str.startsWith("blue"))

return "blue";

else

return "";

}

1. Given a string, return a new string made of 3 copies of the last 2 chars of the original string. The string length will be at least 2.   
     
   extraEnd("Hello") → "lololo"  
   extraEnd("ab") → "ababab"  
   extraEnd("Hi") → "HiHiHi"

Solution:

public String extraEnd(String str) {

String result = str.substring(str.length()-2);

return result + result + result;

}

1. Given a string, if a length 2 substring appears at both its beginning and end, return a string without the substring at the beginning, so "HelloHe" yields "lloHe". The substring may overlap with itself, so "Hi" yields "". Otherwise, return the original string unchanged.   
     
   without2("HelloHe") → "lloHe"  
   without2("HelloHi") → "HelloHi"  
   without2("Hi") → ""

Solution:

public String without2(String str) {

if (str.length() >= 2){

if (str.substring(0,2).equals(str.substring(str.length()-2)))

return str.substring(2,str.length());

else

return str;

}

else

return str;

}

1. Given a string and an int n, return a string made of the first and last n chars from the string. The string length will be at least n.   
     
   nTwice("Hello", 2) → "Helo"  
   nTwice("Chocolate", 3) → "Choate"  
   nTwice("Chocolate", 1) → "Ce"

Solution:

public String nTwice(String str, int n) {

String front = str.substring(0,n);

String back = str.substring(str.length()-n);

return front + back;

}

1. Given two strings, append them together (known as "concatenation") and return the result. However, if the strings are different lengths, omit chars from the longer string so it is the same length as the shorter string. So "Hello" and "Hi" yield "loHi". The strings may be any length.   
     
   minCat("Hello", "Hi") → "loHi"  
   minCat("Hello", "java") → "ellojava"  
   minCat("java", "Hello") → "javaello"

Solution:

public String minCat(String a, String b) {

int aLen = a.length();

int bLen = b.length();

if (aLen > bLen){

return a.substring(aLen - bLen) + b;

} else if (bLen > aLen){

return a + b.substring(bLen - aLen);

} else {

return a + b;

}

}

1. Given a string and an index, return a string length 2 starting at the given index. If the index is too big or too small to define a string length 2, use the first 2 chars. The string length will be at least 2.   
     
   twoChar("java", 0) → "ja"  
   twoChar("java", 2) → "va"  
   twoChar("java", 3) → "ja"

Solution:

public String twoChar(String str, int index) {

if(index <= str.length()-2 && index >= 0)

return str.substring(index, index+2);

return str.substring(0, 2);

}

OR

public String twoChar(String str, int index) {

  if (str.length() <= index + 1 || index < 0)

    return str.substring(0,2);

  else

    return str.substring(index, index + 2);

}

1. Given a string of any length, return a new string where the last 2 chars, if present, are swapped, so "coding" yields "codign".   
     
   lastTwo("coding") → "codign"  
   lastTwo("cat") → "cta"  
   lastTwo("ab") → "ba"

Solution:

public String lastTwo(String str) {

int len = str.length();

if (len < 2)

return str;

else

return str.substring(0,len - 2) + str.charAt(len - 1) + str.charAt(len - 2);

}