Java – warmUp 1

1. Given two temperatures, return true if one is less than 0 and the other is greater than 100.

icyHot(120, -1) → true

icyHot(-1, 120) → true

icyHot(2, 120) → false

Solution:

public boolean icyHot(int temp1, int temp2) {

if ((temp1 < 0) && (temp2 > 100) || (temp2 < 0) && (temp1 > 100)){

return true;

}

else

return false;

}

1. We have two monkeys, a and b, and the parameters aSmile and bSmile indicate if each is smiling. We are in trouble if they are both smiling or if neither of them is smiling. Return true if we are in trouble.

monkeyTrouble(true, true) → true

monkeyTrouble(false, false) → true

monkeyTrouble(true, false) → false

Solution:

public boolean monkeyTrouble(boolean aSmile, boolean bSmile) {

if ((aSmile && bSmile) || (!aSmile && !bSmile)){

return true;

}

else

return false;

}

1. Given two int values, return their sum. Unless the two values are the same, then return double their sum.

sumDouble(1, 2) → 3

sumDouble(3, 2) → 5

sumDouble(2, 2) → 8

Solution:

public int sumDouble(int a, int b) {

if (a != b)

return a+b;

else

return 2\*(a+b);

}

1. Given three int values, a b c, return the largest.

intMax(1, 2, 3) → 3

intMax(1, 3, 2) → 3

intMax(3, 2, 1) → 3

Solution:

public int intMax(int a, int b, int c) {

if ((a>b) && (a>c))

return a;

else if ((b>a) && (b > c))

return b;

else

return c;

}

1. Given a string, return a new string where "not " has been added to the front. However, if the string already begins with "not", return the string unchanged. Note: use .equals() to compare 2 strings.

notString("candy") → "not candy"

notString("x") → "not x"

notString("not bad") → "not bad"

Solution:

public String notString(String str) {

if (str.length()>= 3 && str.substring(0,3).equals("not"))

return str;

else

return "not " + str;

}

1. Given a string, return a new string where the last 3 chars are now in upper case. If the string has less than 3 chars, uppercase whatever is there. Note that str.toUpperCase() returns the uppercase version of a string.

endUp("Hello") → "HeLLO"

endUp("hi there") → "hi thERE"

endUp("hi") → "HI"

Solution:

public String endUp(String str) {

if (str.length()>3){

String front = str.substring(0, str.length()- 3);

String back = str.substring(str.length()- 3);

return (front + back.toUpperCase());

}

else

return (str.toUpperCase());

}

1. Given a string, take the first 2 chars and return the string with the 2 chars added at both the front and back, so "kitten" yields"kikittenki". If the string length is less than 2, use whatever chars are there.

front22("kitten") → "kikittenki"

front22("Ha") → "HaHaHa"

front22("abc") → "ababcab"

Solution:

public String front22(String str) {

if (str.length() < 2)

return str + str + str;

else {

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

return (add + str + add);

}

}

OR

public String front22(String str) {

// First figure the number of chars to take

int take = 2;

if (take > str.length()) {

take = str.length();

}

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

Return front + str + front;

}

1. The parameter weekday is true if it is a weekday, and the parameter vacation is true if we are on vacation. We sleep in if it is not a weekday or we're on vacation. Return true if we sleep in.

sleepIn(false, false) → true

sleepIn(true, false) → false

sleepIn(false, true) → true

Solution:

public boolean sleepIn(boolean weekday, boolean vacation) {

return (!weekday || vacation);

}

1. Given an int n, return the absolute difference between n and 21, except return double the absolute difference if n is over 21.

diff21(19) → 2

diff21(10) → 11

diff21(21) → 0

Solution:

public int diff21(int n) {

if (n <= 21)

return (21 - n);

else

return (n - 21)\*2;

}

1. Given an int n, return true if it is within 10 of 100 or 200. Note: Math.abs(num) computes the absolute value of a number.

nearHundred(93) → true

nearHundred(90) → true

nearHundred(89) → false

Solution:

public boolean nearHundred(int n) {

return(((n >= 90)&& (n <= 110)) || ((n >= 190)&& (n <= 210)));

}

1. Given a non-empty string and an int n, return a new string where the char at index n has been removed. The value of n will be a valid index of a char in the original string (i.e. n will be in the range 0..str.length()-1 inclusive).

missingChar("kitten", 1) → "ktten"

missingChar("kitten", 0) → "itten"

missingChar("kitten", 4) → "kittn"

Solution:

public String missingChar(String str, int n) {

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

String back = str.substring(n+1, str.length());

return front + back;

}

1. We have a loud talking parrot. The "hour" parameter is the current hour time in the range 0..23. We are in trouble if the parrot is talking and the hour is before 7 or after 20. Return true if we are in trouble.

parrotTrouble(true, 6) → true

parrotTrouble(true, 7) → false

parrotTrouble(false, 6) → false

Solution:

public boolean parrotTrouble(boolean talking, int hour) {

return (talking && (hour < 7 || hour > 20));

}

1. Given two non-negative int values, return true if they have the same last digit, such as with 27 and 57. Note that the % "mod" operator computes remainders, so 17 % 10 is 7.   
     
   lastDigit(7, 17) → true  
   lastDigit(6, 17) → false  
   lastDigit(3, 113) → true  
     
   Solution:

public boolean lastDigit(int a, int b) {

return ((a % 10)==(b % 10));

}

1. Given a string, take the last char and return a new string with the last char added at the front and back, so "cat" yields "tcatt". The original string will be length 1 or more.   
     
   backAround("cat") → "tcatt"  
   backAround("Hello") → "oHelloo"  
   backAround("a") → "aaa"  
     
   Solution:  
   public String backAround(String str) {

String last = str.substring(str.length() - 1);

return last + str + last;

}

1. Return true if the given string contains between 1 and 3 'e' chars.   
     
   stringE("Hello") → true  
   stringE("Heelle") → true  
   stringE("Heelele") → false  
     
   Solution: A  
   public boolean stringE(String str) {

int count = 0;

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

if (str.charAt(i) == 'e')

count++;

}

return (count >= 1 && count <=3);

}

OR

public boolean stringE(String str) {

int count = 0;

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

if (str.substring(i, i+1).equals("e"))

count++;

}

return (count >= 1 && count <=3);

}

1. Return true if the given string begins with "mix", except the 'm' can be anything, so "pix", "9ix" .. all count.   
     
   mixStart("mix snacks") → true  
   mixStart("pix snacks") → true  
   mixStart("piz snacks") → false

Solution:  
public boolean mixStart(String str) {

if (str.length() < 3)

return false;

String two = str.substring(1,3);

if (two.equals("ix")){

return true;

} else{

return false;

}

}

1. Given 2 int values, return true if either of them is in the range 10..20 inclusive.   
     
   in1020(12, 99) → true  
   in1020(21, 12) → true  
   in1020(8, 99) → false

Solution:

public boolean in1020(int a, int b) {

if ((a >= 10 && a <= 20 ) || (b >= 10 && b <= 20 )){

return true;

}else{

return false;

}

}

1. Given 2 int values, return whichever value is nearest to the value 10, or return 0 in the event of a tie. Note that Math.abs(n) returns the absolute value of a number.   
     
   close10(8, 13) → 8  
   close10(13, 8) → 8  
   close10(13, 7) → 0

Solution:

public int close10(int a, int b) {

int aDiff = Math.abs(a - 10);

int bDiff = Math.abs(b - 10);

if(aDiff < bDiff){

return a;

} else if (aDiff > bDiff){

return b;

} else {

return 0;

}

}

1. Given a non-empty string and an int N, return the string made starting with char 0, and then every Nth char of the string. So if N is 3, use char 0, 3, 6, ... and so on. N is 1 or more.   
     
   everyNth("Miracle", 2) → "Mrce"  
   everyNth("abcdefg", 2) → "aceg"  
   everyNth("abcdefg", 3) → "adg"

Solution:

public String everyNth(String str, int n) {

String result = "";

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

result = result + str.charAt(i);

}

return result;

}

1. Given 2 positive int values, return the larger value that is in the range 10..20 inclusive, or return 0 if neither is in that range.   
     
   max1020(11, 19) → 19  
   max1020(19, 11) → 19  
   max1020(11, 9) → 11

Solution:

public int max1020(int a, int b) {

// First make it so the bigger value is in a

if (b > a) {

int temp = a;

a = b;

b = temp;

}

// Knowing a is bigger, just check a first

if (a >= 10 && a <= 20)

return a;

if (b >= 10 && b <= 20)

return b;

return 0;

}

1. Given a string, return true if the string starts with "hi" and false otherwise.   
     
   startHi("hi there") → true  
   startHi("hi") → true  
   startHi("hello hi") → false

Solution:

public boolean startHi(String str) {

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

return (str.substring(0,2).equals("hi"));

} else {

return false;

}

}

OR

public boolean startHi(String str) {

// First test if the string is not at least length 2

// (so the substring() below does not go past the end).

if (str.length() < 2) return false;

// Pull out the string of the first two chars

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

// Test if it is equal to "hi"

if (firstTwo.equals("hi")) {

return true;

} else {

return false;

}

// This last part can be shortened to: return(firstTwo.equals("hi"));

}

1. We'll say that a number is "teen" if it is in the range 13..19 inclusive. Given 3 int values, return true if 1 or more of them are teen.   
     
   hasTeen(13, 20, 10) → true  
   hasTeen(20, 19, 10) → true  
   hasTeen(20, 10, 13) → true

Solution:

public boolean hasTeen(int a, int b, int c) {

return ((a > 12 && a < 20) || (b > 12 && b < 20) || (c > 12 && c < 20));

}

1. We'll say that a number is "teen" if it is in the range 13..19 inclusive. Given 2 int values, return true if one or the other is teen, but not both.   
     
   loneTeen(13, 99) → true  
   loneTeen(21, 19) → true  
   loneTeen(13, 13) → false

Solution:

public boolean loneTeen(int a, int b) {

boolean aTeen = (a >= 13 && a <= 19);

boolean bTeen = (b >= 13 && b <= 19);

return (aTeen && !bTeen) || (!aTeen && bTeen);

}

1. Given 2 ints, a and b, return true if one if them is 10 or if their sum is 10.   
     
   makes10(9, 10) → true  
   makes10(9, 9) → false  
   makes10(1, 9) → true

Solution:

public boolean makes10(int a, int b) {

if (a + b == 10)

return true;

else if (a == 10 || b == 10)

return true;

else

return false;

}

1. Given a string, return a new string where the first and last chars have been exchanged.   
     
   frontBack("code") → "eodc"  
   frontBack("a") → "a"  
   frontBack("ab") → "ba"

Solution:

public String frontBack(String str) {

if (str.length() <= 1)

return str;

String middle = str.substring(1,str.length()-1);

return (str.charAt(str.length()-1)+ middle + str.charAt(0));

}

1. Given a string, if the string "del" appears starting at index 1, return a string where that "del" has been deleted. Otherwise, return the string unchanged.   
     
   delDel("adelbc") → "abc"  
   delDel("adelHello") → "aHello"  
   delDel("adedbc") → "adedbc"

Solution:

public String delDel(String str) {

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

return (str.charAt(0) + str.substring(4,str.length()));

else

return str;

}

OR

public String delDel(String str) {

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

// First char + rest of string starting at 4

return str.substring(0, 1) + str.substring(4);

}

// Otherwise return the original string.

return str;

}

1. Given a string, return a string made of the first 2 chars (if present), however include first char only if it is 'o' and include the second only if it is 'z', so "ozymandias" yields "oz".   
     
   startOz("ozymandias") → "oz"  
   startOz("bzoo") → "z"  
   startOz("oxx") → "o"

Solution:

public String startOz(String str) {

String result = "";

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

result = result + str.charAt(0);

}

if (str.length() >= 2 && str.charAt(1)=='z'){

result = result + str.charAt(1);

}

return result;

}

1. Return true if the given non-negative number is a multiple of 3 or a multiple of 5. Use the % "mod" operator   
     
   or35(3) → true  
   or35(10) → true  
   or35(8) → false

Solution:

public boolean or35(int n) {

return (n % 3 == 0) || (n % 5 == 0);

}

1. Given 2 int values, return true if one is negative and one is positive. Except if the parameter "negative" is true, then return true only if both are negative.   
     
   posNeg(1, -1, false) → true  
   posNeg(-1, 1, false) → true  
   posNeg(-4, -5, true) → true

Solution:

public boolean posNeg(int a, int b, boolean negative) {

if (negative){

return (a < 0 && b < 0);

} else {

return ((a > 0 && b < 0)||(a < 0 && b > 0));

}

}

1. Given 2 int values, return true if they are both in the range 30..40 inclusive, or they are both in the range 40..50 inclusive.   
     
   in3050(30, 31) → true  
   in3050(30, 41) → false  
   in3050(40, 50) → true

Solution:

public boolean in3050(int a, int b) {

return (((a >= 30 && a <= 40) && (b >= 30 && b <= 40)) || ((a >= 40 && a <= 50) && (b >= 40 && b <= 50)));

}

1. Given a string, we'll say that the front is the first 3 chars of the string. If the string length is less than 3, the front is whatever is there. Return a new string which is 3 copies of the front.   
     
   front3("Java") → "JavJavJav"  
   front3("Chocolate") → "ChoChoCho"  
   front3("abc") → "abcabcabc"

Solution:

public String front3(String str) {

String front;

if (str.length() >= 3) {

front = str.substring(0,3);

} else {

front = str;

}

return front + front + front;

}