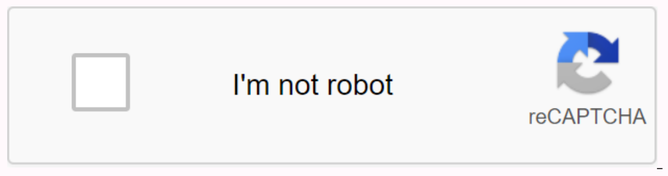

Continue

Codingbat java recursion answers

Recursion-1 chanceBasic recursion problems. Recursion strategy: the first test is one or two basic cases that are so simple, the answer can be returned immediately. Otherwise, make a recursive call for a smaller case (i.e. a case that is a step in the base case). Let's say the recursive call works properly and correct what it's returning to the response. Java HelpMisc Code Practice Coding BAT RESPONSES moves to a new and better site, please click here to find solutions to any JavaBAT problem and learn from my mistakes!!!! This section contains the following questions: factorial, bunnyEars, fibonacci, bunnyEars2, triangle, sumDigits, count7, count8, powerN, countX, countHi, changeXY, changePi, noX, array6, array11, array220, allStar, pairStar, endX countPairs, countAbc, count11, stringClean, countHi2, parenBit, nestParen, strCount, strCopies, strDist. factorial public int factorial(int n) { if(n == 1) { return 1; } else { return factorial(n-1) * n; } } bunnyEars public int bunnyEars(int bunnies) { if(bunnies == 0) return 0; else return bunnies + bunnies; } fibonacci public int fibonacci(int n) { if(n == 0) return 0; if(n == 1) return 1; else { return fibonacci(n - 2) + fibonacci(n - 1); } } bunnyEars2 public int bunnyEars2(int bunnies) { if(bunnies == 0) return 0; else return bunnies + bunnies + (bunnies / 2); } triangle public int triangle(int rows) { if (rows == 0) return 0; if (rows == 1) return 1; else return triangle(rows - 1) + rows; } sumDigits public int sumDigits(int n) { int sum; if (n == 0) return 0; else { sum = n % 10; n = n / 10; return sumDigits(n) + sum; } } count7 public int count7(int n) { int sum; if(n == 0) return 0; else { sum = n % 10; n = n / 10; if (sum == 7) return 1 + count7(n); else return count7(n); } } count8 public int count8(int n) { int sum; if (n == 0) 0 return; other { sum = n % 10; n = n / 10; ha (amount == 8 & n % 10 == 8) 2 + count8(n); if (amount == 8) 1 + count8(n); else return count8(n); } } powerN public int powerN(int base, int n) { if (n == 1) return base; if (n == 0) return 0; else return base * powerN(base, n - 1); } countX public int countX(String str) { int result = 0; if(str.length() == 0) { return 0; } if(str.charAt(0) == 'x') { result++; } result += countX(str.substring(1, str.length())); return result; } countHi public int countHi(String) { int result = 0; if(str.length() <= 2) { if(str.equals(hi)) { return 1; } return 0; } if(str.substring(0, 2).equals(hi)) { result++; str = str.sub String(2; str.length()); } else { str = str.substring(1, str.length()); } result += countHi(str); return result; } changeXY public String changeXY(String str) { String result = ; if(str.length() == 0) { return ; } if(str.charAt(0) == 'x') { result = y; } else { result = Character.toString(str.charAt(0)); } return result + changeXY(str.substring(1, str.length())); } String changePi(String str) { String result = ; if(str.length() < 2) { return str; } if(str.charAt(0) == 'p' & str.charAt(1) == 'i') { result = 3.14; str = str.substring(2, str.length()); } else { result = Character.toString(str.charAt(0)); str = str.substring(1, str.length()); } return result + changePi(str); } noX public String noX(String str) { String result = ; if(str.length() == 0) { return ; } if(str.charAt(0) == 'x') { result = ; } else { result = Character.toString(str.charAt(0)); } return result + noX(str.substring(1 , str.length())); } array6 nyilvános logikai tömb6(int[] nums, int index) { if(index >= nums.length) { return false; } else if(nums[index] == 6) { return true; } else { return array6(nums, index + 1); } } array11 public int array11(int[] nums, int index) { int result = 0; if(index >= nums.length) { return 0; } if(nums[index] == 11) { result++; } result += array11(nums; index + 1); return result; } array220 public boolean array220(int[] nums , int index) { if(index >= nums.length - 1) { return false; } else if(nums[index] * 10 == nums[index + 1]) { return true; } else { return array220(nums, index + 1); } } allStar nyilvános karakterlánc allStar(String str) { String = ; if(str.length() == 1 || str.length() == 0) { return str; } else { result += (Character.toString(str.charAt(0)) + *); } eredmény += allStar(str.substring(1, str.length())); visszatérési eredmény; } pairStar nyilvános karakterlánc-pairCsillag(String str) { String result = ; if(str.length() <=1) { return str; } if(str.charAt(0) == str.charAt(1)) { result += (Character.toString(str.charAt(0)) + *); } else { result += Character.toString(str.charAt(0)); } result += pairStar(str.substring(1, str.length()); visszatérési eredmény; } endX nyilvános karakterlánc endX(Karakterlánc str) { Karakterlánc eredménye = ; int x = countX(str); eredmény = removeX(str); eredmény += addX(x); visszatérési eredmény; } public int countX(String str) { int x = 0; if(str.length() <= 0) { return 0; } if(str.charAt(0) == 'x') { x++; } x += countX(str.substring(1, str.length())); return x; } public String removeX(str) { String = String = eredmény; if(str.length() <= 0) { return ; } máshol, ha(str.charAt(0) == 'x') { eredmény += ;} else { result += Character.toString(str.charAt(0)); } result += removeX(str.substring(1 , str.length())); visszatérési eredmény; } nyilvános karakterlánc-addX(int num) { if(num < 1) { return ; } if(num == 1) { return x; } vissza x + addX(num - 1); } countPairs public int countPairs(String str) { int result = 0; if(str.length() <= 2) { return 0; } if(str.charAt(0) == str.charA(2)) { result++; } result += countPairs(str.substring(1, str.length())); return result; } countAbc public int countAbc(String str) { int result = 0; if(str.length() <= 2) { return 0; } if(str.substring(0 , 3).equals(aba) || str.substring(0, 3).equals(abc)) { } result += countAbc(str.substring(1, str.length())) return result} count1 public public count11(Karakterlánc str) { int result = 0; ha(str.length() <= 1) { return 0; } if(str.substring(0; 2).equals(11) || str.substring(0, 2).equals(11)) { result++; str = str.substring(2, str.length()); } else { str = str.substring(1, str.length(.length()); } result += count11(str); return result; } stringClean public String StringClean(String str) { if(str.length() <= 1) { return str; } Karakterlánc eredménye = , c = Karakter.toString(str.charAt(0)), n = ; do { if(n.equals(c)) { str = str.substring(1; str.length()); if(str.length() <= 1) { break; } } n = Character.toString(str.charAt(1)); } while(n.equals(c) & str.length() > 1); if(str.length() < 1) { } else { str = str.substring(1, str.length()); } eredmény += c; eredmény += stringClean(str); visszatérési eredmény; } countHi2 public int countHi2(String str) { int result = 0; if(str.length() < 2) { return 0; } else if(str.length() == 2) { if(str.equals(hi)) { return 1; } else { return 0; } } else { if(str.charAt(0) == 'x') { if(str.length() <= 3) { return 0; } else if(str.substring(1, 3).equals(hi)) { str = str.substring(3, str.length()); } else { str = str.substring(1, str.length()); } } else { if(str.substring(0, 2).equals(hi)) { result++; str = str.substring(2 , str.length()); } else { str= str.substring(1, str.length()); } result += countHi2(str); return result; } parenBit public String parenBit(String str) { String result = ; if(str.charAt(0) == '(') { if(str.charAt(str.length() - 1) == ')') { return str; } else { return parenBit(str.substring(0, str.length() - 1)); } else { return parenBit(str.substring(1, str.length()) } } nestParen nyilvános logikai nestParen(Karakterlánc str) { Karakterlánc eredmény = ; if(str.length() == 0) { return true; } if(str.length() < 1) { return false; } if(str.charAt(0) == '(') {if(str.charAt(str.length() - 1) == ')') { return nestParen(str.substring(1, str.length() -1)); } return false} } } strCount public int strCount(String str; String sub) { int result = 0; if(str.length() < sub.length()) { return 0; } else if(str.length() == sub.length() & str.equals(sub)) { return 1; } else if(str.substring(0, sub.length(.). equals(sub)) { result++; str = str.substring(sub.length(, str.length()); } else { str = str.substring(1, str.length()); } result += strCount(str, sub); return result; } strCopies public boolean strCopies(String str, String sub, int n) { if(strCount(str, sub) == n) { return duration; } strCopies public boolean strCopies(String str, String sub, int n) { if(strCount(str, sub) == n) { return. } else { return false; } } public int strCount(String str, String sub) { int result = 0; if(str.length() < sub.length()) { return 0; } else if(str.length() == sub.length() & str.length(sub)) { return 1; } else if(str.substring(0, sub.length(.)). equals(sub)) { result++; } str= str.substring(1; str.length()); result += sub); return result; } strDist public int strDist(String str, String sub) { if(str.length() == 1 & str.equals(sub)) { return 1; } else else < al.length() || str.length() <= 1) { return 0; } other { if(str.charAt(0) == sub.charAt(0) &gta str.charAt(0) == str.charAt(str.length() - sub.length()) &; str.substring(str.length(str.length(str.length(.str.length(.0) == str.charAt(str.length() - sub.length()) str.length(str.length(str.length (sub.length().equals(sub) &gt.substring(str.length() - sub.length()).equals(sub)) { return str.length(); } other { if(str.substring(0, sub.length().equals(sub)) { return strDist(str.substring(0; str.length(1), sub); } más ha(str.substring(str.length() - sub.length().equals(sub)) { return strDist(str.substring(1 , str.length()), sub); } return strDist(str.substring(1, str.length(1, sub} Ez a resz a következő kérdéseket tartalmazza: faktoriális, bunnyHears, fibonacci, bunnyHears2, hármszög, sumDigits, count7, count8, powerN, countX, countHi, changeXY, changePi, noX, array6, array11, array220, allStar, pairStar, endX countPairs, countAbc, count11, stringClean, countHi2, parenBit, nestParent, strCount, strCode, strDis strDis.