17-09-16

***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("Java", 1) → "Jva"***

***missingChar("Android", 0) → "ndroid"***

import java.util.\*;

public class solution2a{

public static String missingChar(String str, int n) {

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

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

return front + back;

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String a=sc.next();

int n=sc.nextInt();

System.out.println(missingChar(a,n));

}

}

***Given a string, take the last char and return a new string with the last char added at the front and back. The original string will be length 1 or more. backAround("Hello") → "oHelloo"***

import java.util.\*;

public class solution2b

{

public static String backAround(String str)

{

int n=str.length();

char s=str.charAt(n-1);

return s+str+s;

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String str=sc.next();

System.out.println(backAround(str));

}

}

***Return true if the given string contains between 1 and 3 'e' chars.***

***containE(“Hello”) “true”***

***containE(“Helloeee”) “false”***

***containE(“World”) “false”***

import java.util.\*;

public class solution2c

{

public static boolean containE(String str)

{

int cnt=0;

int n=str.length();

for(int i=0;i<n;i++)

{

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

cnt++;

}

if(cnt>=1&&cnt<=3)

return true;

else

return false;

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String str=sc.next();

System.out.println(containE(str));

}

}

***Given an even length first string, such as "<<>>", and a second string “word”, return a new string where the second string is in the middle of the first string, e.g. "<<word>>".***

***embedWord("<<>>", "Hello") "<<Hello>>"***

***embedWord("<>", "HTML") "<HTML>"***

***embedWord("[]", "1,2,3,4,5") "[1,2,3,4,5]"***

***embedWord("{[]}", "a:1, b:2, c:3") "{[a:1, b:2, c:3]}"***

import java.util.\*;

public class solution2d

{

public static String embedWord(String s1,String s2)

{

int n=s1.length();

return (s1.substring(0, n/2) + s2 + s1.substring((n/2), n));

}

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String str=sc.next();

String str2=sc.next();

System.out.println(embedWord(str,str2));

}

}

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

***makeTags("i", "Hello") "<i>Hello</i>"***

***makeTags("h1", "Android") "<h1>Android</h1>"***

import java.util.\*;

public class solution2e{

public static String makeTags(String tag, String word)

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

public static void main(String args[])

{

Scanner sc=new Scanner(System.in);

String str=sc.next();

String str2=sc.next();

System.out.println(makeTags(str,str2));

}

}