********** OUTPUT **********

Expected	Run		
changePi("xpix") → "x3.14x"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("pipi") → "3.143.14"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("pip") → "3.14p"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("pi") → "3.14"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("hip") → "hip"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("p") → "p"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("x") → "x"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("") → ""	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("pixx") → "3.14xx"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
changePi("xyzzy") → "xyzzy"	StringIndexOutOfBoundsException: String index out of range: -1 (line:18)	X	
other tests		X	

*** INITIAL CODE (CONSISTENT WITH OUTPUT ABOVE) ***

// this first line is part of the pre-populated solution and can not be modified
// since the signature can not be changed, the parameter is limited to a single string. // In order for recursion to work, it creates a tough scenario since there is no reference to an index… And a recursive call to the method will not keep accountability of any incremented variables..
// So on this instance, I did some research on the internet to understand a logistical way…
// The code presented here is a personal attempt, and I have tried demonstrate my rationale

```
public String changePi(String str) {

 String newStart=null;          // this will be used to pass into the recursive
 call int pos;   // this will hold index location of “pi”
 if (str.indexOf("pi")!=-1)     // if it finds a match of pi
 {
  pos=str.indexOf("pi");      // this will hold index location of “pi”

  if (pos<=str.length()-2)     // this checks that index of pi can not appear no further in string
                                than following abcdpid
                               // pos is zero index… so in this instance, its value would be 4.
                               //length of string would be 7.. Hence it will not create another
                                newStart substring for a recursive call (4 < (7-2)
```

```
                                        //However abcdpide is valid to perform another recursive call…
pos = 4 and length string = 8          (4 < (8-2))
{
   newStart = str.substring(pos+2);
// the string holds the new start location to check again for pi

   return "3.14" + changePi(newStart);
// this will now return 3.14 as per the challenge and then continue recursive call and search
again for pi

   }
  }
  newStart=str.substring(1);     // this is outside of the loop. It has not found "pi" in the string.
// so on this instance, it will continue from an index location further into the string. Since this
level of information can not be passed as a parameter, the String has to be truncated.

  return str.charAt(0) + changePi(newStart); // this will return the first character in the
original string (since it has not found pi) and will continue remainder of the string….

}
```

I have tried to fix IndexOutofBound exception since it can be realized that I had no measure to determine when to stop recursive calls.

```
if (newStart.length()>=2)
// now there is control that if the truncated string that is being passed into the method is at
least two characters in length…

{
newStart=str.substring(1);      //the newStart can now be set to the second character based
on zero notation
  return str.charAt(0) + changePi(newStart);
// this will now return the character (which does not meet the "pi" condition and start
recursively checking from the next character inline
}

return str;        //if the string is 1 character or less, it can not be evaluated for "pi" substring,
hence entire string is returned……

}
```

My result was much improved:

Expected	Run		
changePi("xpix") → "x3.14x"	"3.14x"	X	
changePi("pipi") → "3.143.14"	"3.143.14"	OK	
changePi("pip") → "3.14p"	"3.14p"	OK	
changePi("pi") → "3.14"	"3.14"	OK	
changePi("hip") → "hip"	NullPointerException (line:24)	X	
changePi("p") → "p"	"p"	OK	
changePi("x") → "x"	"x"	OK	
changePi("") → ""	""	OK	
changePi("pixx") → "3.14xx"	NullPointerException (line:24)	X	
changePi("xyzzy") → "xyzzy"	NullPointerException (line:24)	X	
other tests		X	

Your progress graph for this problem

I found only technique to remove nullpointer exception was related to my initialization of:

String newStart= null;

I adjusted this as follows:

String newStart= " ";

However the execution was not entirely complete, but extremely close:

Expected	Run		
changePi("xpix") → "x3.14x"	"3.14x"	X	
changePi("pipi") → "3.143.14"	"3.143.14"	OK	
changePi("pip") → "3.14p"	"3.14p"	OK	
changePi("pi") → "3.14"	"3.14"	OK	
changePi("hip") → "hip"	"hip"	OK	
changePi("p") → "p"	"p"	OK	
changePi("x") → "x"	"x"	OK	
changePi("") → ""	""	OK	
changePi("pixx") → "3.14xx"	"3.14xx"	OK	
changePi("xyzzy") → "xyzzy"	"xyzzy"	OK	
other tests		X	

*** NEW CODE (CONSISTENT WITH OUTPUT ABOVE) ***

```
public String changePi(String str) {

  String newStart= "";
  int pos;

  if (str.length()<2)
  {
    return str;
  }

    if (str.indexOf("pi")!=-1)
  {
    pos=str.indexOf("pi");

    if (pos<=str.length()-2)
    {
    newStart = str.substring(pos+2);

    return "3.14" + changePi(newStart);

    }
  }

if (newStart.length()>=2)
{
  newStart=str.substring(1);
  return str.charAt(0) + changePi(newStart);
}

return str;

}
```

Expected	Run		
changePi("xpix") → "x3.14x"	"x3.14x"	OK	
changePi("pipi") → "3.143.14"	"3.143.14"	OK	
changePi("pip") → "3.14p"	"3.14p"	OK	
changePi("pi") → "3.14"	"3.14"	OK	
changePi("hip") → "hip"	"hip"	OK	
changePi("p") → "p"	"p"	OK	
changePi("x") → "x"	"x"	OK	
changePi("") → ""	""	OK	
changePi("pixx") → "3.14xx"	"3.14xx"	OK	
changePi("xyzzy") → "xyzzy"	"xyzzy"	OK	
other tests		OK	

*** NEW CODE (CONSISTENT WITH OUTPUT ABOVE) ***

I managed to find a solution to this problem… It was very tricky and perhaps without the green and red flags, I would have never figured out how to solve this…

```
public String changePi(String str) {

 String temp= "";
 int pos;

 if (str.length()<2)
 {
  return str;
 }

 if (str.charAt(0)=='p' && str.charAt(1)=='i')
 {
  temp=str.substring(2);
  return 3.14 + changePi(temp);
 }

 return str.charAt(0)+changePi(str.substring(1));
```