

CompareTo method is going to compare the string in a lexicographical (dictionary order) manner. The return type is integer.

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println("A".compareTo("A")); //0
8         System.out.println("A".compareTo("C")); //-2
9         System.out.println("C".compareTo("A")); //2
10        System.out.println("L".compareTo("Q")); //-5
11        System.out.println("Z".compareTo("R"));
12    }
13    public static void main(String[] args)
14    {
15        new ClassA().meth1();
16    }
17 }
```

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     void meth1()
6     {
7         System.out.println("A".compareTo("A")); //0
8         System.out.println("A".compareTo("C")); //-2
9         System.out.println("C".compareTo("A")); //2
10        System.out.println("L".compareTo("Q")); //-5
11        System.out.println("Z".compareTo("R")); //8
12        System.out.println("A".compareTo("a")); //-32
13        System.out.println("A".compareToIgnoreCase("a")); //0
14    }
15    public static void main(String[] args)
16    {
17        new ClassA().meth1();
18    }
19 }
```

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!"
```


```
public String helloName(String name)  
{  
    // write ur logic here  
}
```

```
1 package com.pack1;  
2  
3 public class ClassA  
4 {  
5     public String helloName(String name)  
6     {  
7         //return "Hello ".concat(name).concat("!");  
8         return "Hello ".concat(name)+"!";  
9     }  
10    public static void main(String[] args)  
11    {  
12        ClassA aobj=new ClassA();  
13        String result=aobj.helloName("Java");  
14        System.out.println(result);// Hello Java!  
15    }  
16 }
```

Given two strings, a and b, return the result of putting them together in the order ab-ba, e.g. "Hi" and "Bye" returns "HiByeByeHi".

```
makeAbba("Hi", "Bye") ? "HiByeByeHi"  
makeAbba("Yo", "Alice") ? "YoAliceAliceYo"  
makeAbba("What", "Up") ? "WhatUpUpWhat"
```

```
public String makeAbba(String a, String b)  
{  
    // write ur logic here  
}
```



```
1 package com.pack1;  
2  
3 public class ClassA  
4 {  
5     public String makeAbba(String a, String b)  
6     {  
7         return a+b+b+a;  
8     }  
9  
10    public static void main(String[] args)  
11    {  
12        ClassA aobj=new ClassA();  
13        String result=aobj.makeAbba("Hi", "Bye");  
14        System.out.println(result); //HiByeByeHi  
15    }  
16 }
```

Palindrome

If we read from forward to back and back to front the meaning is the same.

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     void checkName(String name)
6     {
7         String rev="";
8         for(int i=name.length()-1;i>=0;i--)
9         {
10             rev=rev+name.charAt(i);
11         }
12         System.out.println("name : "+name);
13         System.out.println("rev : "+rev);
14
15         if(name.equals(rev))
16             System.out.println(name+" is palindrome");
17         else
18             System.out.println(name+" is NOT a palindrome");
19     }
20     public static void main(String[] args)
21     {
22         ClassA aobj=new ClassA();
23         aobj.checkName("Madam");
24     }
```

name : Madam
rev : madaM
Madam is NOT a palindrome

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     void checkName(String name)
6     {
7         String rev="";
8         for(int i=name.length()-1;i>=0;i--)
9         {
10             rev=rev+name.charAt(i);
11         }
12         System.out.println("name : "+name);
13         System.out.println("rev : "+rev);
14
15         if(name.equalsIgnoreCase(rev))
16             System.out.println(name+" is palindrome");
17         else
18             System.out.println(name+" is NOT a palindrome");
19     }
20     public static void main(String[] args)
21     {
22         ClassA aobj=new ClassA();
23         aobj.checkName("Madam");
24     }
```

name : Madam
rev : madaM
Madam is palindrome

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>>".

makeOutWord("<<>>", "Yay") ? "<<Yay>>"

makeOutWord("<<>>", "WooHoo") ? "<<WooHoo>>"

makeOutWord("[[]]", "word") ? "[[word]]"

```
public String makeOutWord(String out, String word)
{
    // write ur logic here
}
```

```
2
3 public class ClassA
4
5     public String makeOutWord(String out, String word)
6     {
7         return out.charAt(0)+out.charAt(1)+word+out.charAt(2)+out.charAt(3);
8     }
9
10    public static void main(String[] args)
11    {
12        ClassA aobj=new ClassA();
13        String result=aobj.makeOutWord("<<>>", "Java");
14        System.out.println(result);//<<Java>>
15    }
```

```
2
3 public class ClassA
4 {
5     public String makeOutWord(String out, String word)
6     {
7         return out.substring(0, 2)+word+out.substring(2);
8     }
9
10    public static void main(String[] args)
11    {
12        ClassA aobj=new ClassA();
13        String result=aobj.makeOutWord("<<>>", "Java");
14        System.out.println(result);//<<Java>>
15    }
16 }
```


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"

```
public String extraEnd(String str)
{
    // Write your logic here
}
```

```
1 package com.pack1;
2
3 public class ClassA
4 {
5     public String extraEnd(String str)
6     {
7         String s=str.substring(str.length()-2);
8         return s+s+s;
9     }
10    public static void main(String[] args)
11    {
12        ClassA aobj=new ClassA();
13        String result=aobj.extraEnd("NareshIt");
14        System.out.println(result); // ItItIt
15    }
16 }
```

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 "".

```
firstTwo("Hello") ? "He"
firstTwo("abcdefg") ? "ab"
firstTwo("ab") ? "ab"
```

```
public String firstTwo(String str) {
    // Write your logic here
}

1 package com.pack1;
2
3 public class ClassA
4 {
5     public String firstTwo(String str)
6     {
7         int len=str.length();
8         if(len>=2)
9             return str.substring(0,2);
10        return str;
11    }
12    public static void main(String[] args)
13    {
14        ClassA aobj=new ClassA();
15        String result=aobj.firstTwo("");
16        System.out.println(result); //Na
17    }
18 }
```

Given a string, return a string where for every char in the original, there are two chars.

```
doubleChar("The") ? "TThhee"  
doubleChar("AAbb") ? "AAAAAbbbb"  
doubleChar("Hi-There") ? "HHii--TThheerree"
```

```
public String doubleChar(String str)  
{  
    // Write ur logic here  
}
```

```
1 package com.pack1;  
2  
3 public class ClassA  
4 {  
5     public String doubleChar(String str)  
6     {  
7         String s="";  
8         for(int i=0;i<=str.length()-1;i++)  
9         {  
10             s=s+str.charAt(i)+str.charAt(i);  
11         }  
12         return s;  
13     }  
14 }  
15  
16 public static void main(String[] args)  
17 {  
18     ClassA aobj=new ClassA();  
19     String result=aobj.doubleChar("Hi-There");  
20     System.out.println(result);  
21 }  
22 }
```


Assignments

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!"

```
public String helloName(String name)
{
    // write ur logic here
}
```

Given two strings, a and b, return the result of putting them together in the order ab-ba, e.g. "Hi" and "Bye" returns "HiByeByeHi".

makeAbba("Hi", "Bye") ? "HiByeByeHi"

makeAbba("Yo", "Alice") ? "YoAliceAliceYo"

makeAbba("What", "Up") ? "WhatUpUpWhat"

```
public String makeAbba(String a, String b)
{
    // write ur logic here
}
```

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>>".

makeOutWord("<<>>", "Yay") ? "<<Yay>>"

makeOutWord("<<>>", "WooHoo") ? "<<WooHoo>>"

makeOutWord("[[]]", "word") ? "[[word]]"

```
public String makeOutWord(String out, String word)
{
    // write ur logic here
}
```

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"

```
public String extraEnd(String str)
{
```

```
// Write your logic here  
}
```

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 "".

```
firstTwo("Hello") ? "He"  
firstTwo("ABCDEFGH") ? "AB"  
firstTwo("AB") ? "AB"
```

```
public String firstTwo(String str)  
{  
    // Write your logic here  
}
```

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"
```

```
public String firstHalf(String str)
```

```
{  
    // Write your logic here  
}
```

Given a string, return a string where for every char in the original, there are two chars

doubleChar("The") ? "TThhee"

doubleChar("AAbb") ? "AAAAbbbb"

doubleChar("Hi-There") ? "HHii--TThheerree"

```
public String doubleChar(String str)  
{  
    // Write ur logic here  
}
```

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

```
public boolean frontAgain(String str)  
{
```

```
// Write your logic here
```

```
}
```

Return the number of times that the string "hi" appears anywhere in the given string.

countHi("abc hi ho") ? 1

countHi("ABChi hi") ? 2

countHi("hihi") ? 2

```
public int countHi(String str)
```

```
{
```

```
    // Write your logic here
```

```
}
```

--

Return the number of times that the string "code" appears anywhere in the given string, except we'll accept any letter for the 'd', so "cope" and "cooe" also count.

countCode("aaacodebbb") ? 1


```
countCode("codexxcode") ? 2  
countCode("cozexxcope") ? 2
```

```
public int countCode(String str)  
{  
    // Write your logic here  
}
```