```java
import java.util.concurrent.*;
import java.util.function.*;
import java.util.regex.*;
import java.util.stream.*;
import static java.util.stream.Collectors.joining;
import static java.util.stream.Collectors.toList;

class Result {

    /*
     * Complete the 'findObfuscateMessage' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts STRING message as parameter.
     */

    public static String findObfuscateMessage(String message) {
    // Write your code here

    }

}

public class Solution {…
```

1. Code Question 1

Amazon developers are creating a new security module for Amazon Web Services. One of its functions is to obfuscate incoming messages. The obfuscation method is to select some substring of the message and right rotate that substring by 1 position one time only. The substring is selected in such a way that the resulting obfuscated message is alphabetically the smallest message that can be formed using this operation.

Here, right rotating any string *a* by 1 position is to shift every character of string *a* in the right direction by 1 position and remove the last character and append it to the beginning of string *a*. For example, *a* = "ahssd", after right rotation by 1 position string *a* changes to "dahss".

Given the input *message* as a string, find the obfuscated message.

Note: A string *x* is alphabetically smaller than string *y*, if either *x* is a prefix of *y* (and $x \neq y$), or there exists such i ($0 \leq i < min(|x|, |y|)$), that $x[i] < y[i]$, and for any j ($0 \leq j < i$) $x[j] = y[j]$. Here, $|a|$ denotes the length of the string *a*.

Example

message = "aahhab"

These are some of the possible strings.

Starting Index	Ending Index
2	5
0	2
1	2
2	4

Rotating the substring "hha" produces "ahh". Replace "hha" in the original string to get "aaahhb", the alphabetically smallest message that can be formed. Return "aaahhb".

Function Description

Complete the function *findObfuscateMessage* in the editor below.

findObfuscateMessage has the following parameter:

string message: the input message

Returns

string: the encrypted message

1. Code Question 1

Amazon developers are creating a new security module for Amazon Web Services. One of its functions is to obfuscate incoming messages. The obfuscation method is to select some substring of the message and right rotate that substring by 1 position one time only. The substring is selected in such a way that the resulting obfuscated message is alphabetically the smallest message that can be formed using this operation.

Here, right rotating any string *a* by 1 position is to shift every character of string *a* in the right direction by 1 position and remove the last character and append it to the beginning of string *a*. For example, *a* = "ahssd", after right rotation by 1 position string *a* changes to "dahss".

Given the input *message* as a string, find the obfuscated message.

Note: A string *x* is alphabetically smaller than string *y*, if either *x* is a prefix of *y* (and $x \neq y$), or there exists such *i* ($0 \leq i < min(|x|, |y|)$), that $x[i] < y[i]$, and for any *j* ($0 \leq j < i$) $x[j] = y[j]$. Here, $|a|$ denotes the length of the string *a*.

Example

message = "aahhab"

These are some of the possible strings.

Starting Index	Ending Index
2	5
0	2
1	2
2	4

Rotating the substring "hha" produces "ahh". Replace "hha" in the original string to get "aaahhb", the alphabetically smallest message that can be formed. Return "aaahhb".

Function Description

Complete the function *findObfuscateMessage* in the editor below.

findObfuscateMessage has the following parameter:

string message: the input message

Returns

string: the encrypted message

Constraints

- $1 \leq |message| \leq 10^5$
- The string *message* contains only lower case English letters.

▼ Input Format For Custom Testing

The first and the only line contains a string, *message*

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       --------
bbasa →     message = "bbasa"
```

Sample Output

```
abbas
```

Explanation

message can be converted into strings "abbsa", "bbaas", etc. It is optimal to choose the substring "bbasa" which rotates to "abbas".

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       --------
aabc  →     message = "aabc"
```

string, *message*

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       --------
bbasa →     message = "bbasa"
```

Sample Output

```
abbas
```

Explanation

message can be converted into strings "abbsa", "bbaas", etc. It is optimal to choose the substring "bbasa" which rotates to "abbas".

▼ Sample Case 1

Sample Input For Custom Testing

```
STDIN       FUNCTION
-----       --------
aabc  →     message = "aabc"
```

Sample Output

```
aabc
```

Explanation

The string "aabc" is already the lexicographically smallest string. We can choose to rotate any 1 letter substring of the given *message*.

2. Code Question 2

Developers at Amazon are working on a text generation utility for one of their new products.

Currently, the utility generates only special strings. A string is special if there are no matching adjacent characters. Given a string *s* of length *n*, generate a special string of length *n* that is lexicographically greater than *s*. If multiple such special strings are possible, then return the lexicographically smallest string among them.

Notes:

- Special String: A string is special if there are no two adjacent characters that are the same.
- Lexicographical Order: This is a generalization of the way words are alphabetically ordered in dictionaries. For example, "abc" is lexicographically smaller than "abd" because 'c' comes before 'd' in the alphabet.

A string a is lexicographically smaller than a string b if and only if one of the following holds:

- a is a prefix of b, but a is not equal to b. For example, "abc" is smaller than "abcd".
- In the first position where a and b differ, the character in a comes before the character in b in the alphabet. For example, "abc" is smaller than "abd" because 'c' comes before 'd'.

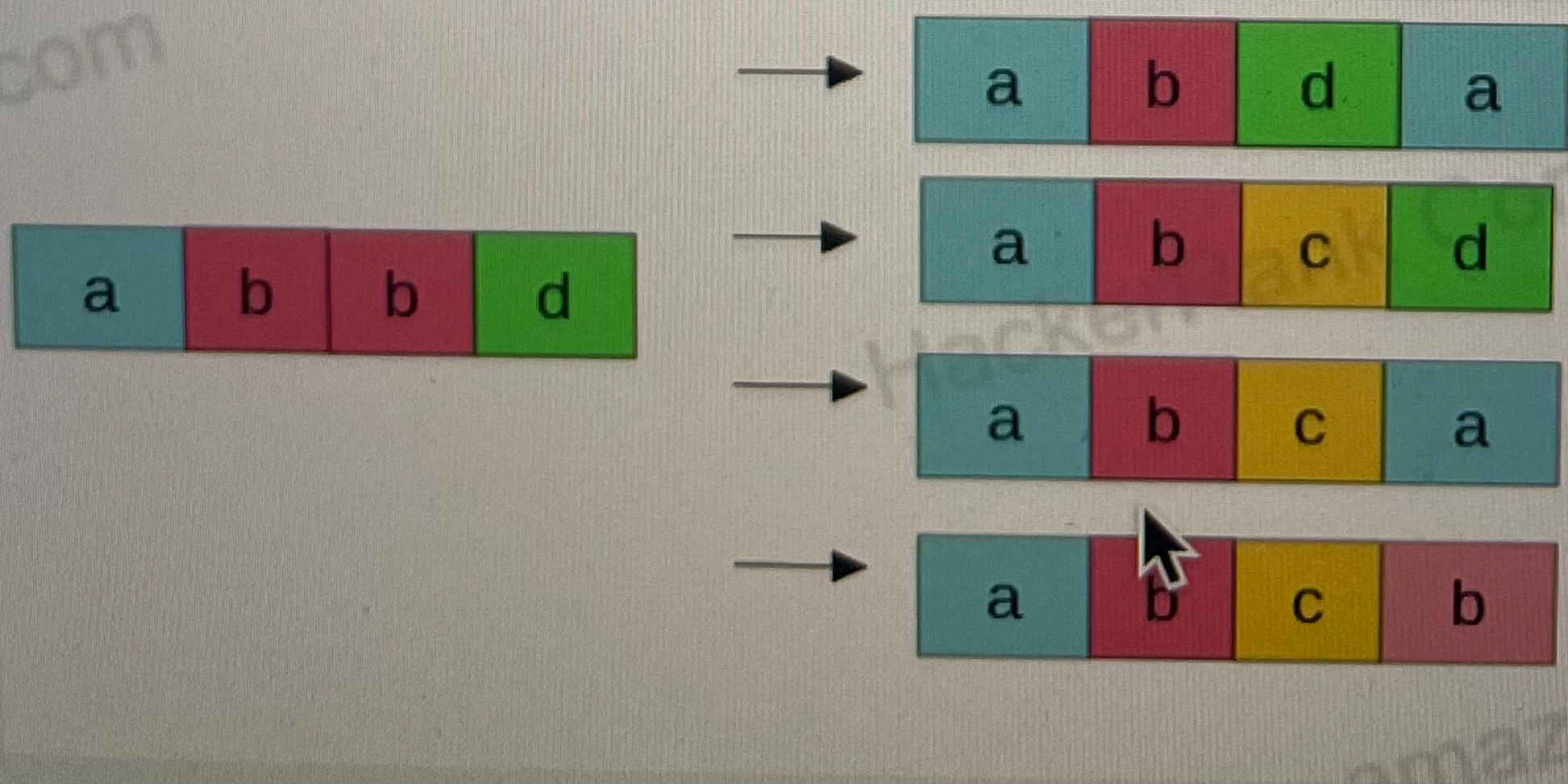
Important Considerations:

- If the character is 'z', it is the last character in the alphabet and cannot be increased further. The string should not wrap around to 'a' after 'z'.
- The output string must not have any adjacent characters that are the same.

Example

Suppose *s* = *"abbd"*.

Some of the special strings that are lexicographically greater than s are shown.



The lexicographically smallest special string that is greater than "abbd" is "abca".

Function Description

Complete the function *getSpecialString* in the editor below.

getSpecialString has the following parameter:

s: the input string

Returns

string: the lexicographically smallest string that is greater than *s*. If no such special string exists, return "-1".

Constraints:

- $1 \leq |s| \leq 10^6$
- *s* consists of lowercase English letters only.

Constraints:

- $1 \leq |s| \leq 10^6$
- s consists of lowercase English letters only.

▼ Input Format For Custom Testing

The only line of input contains a string s.

▼ Sample Case 0

Sample Input For Custom Testing

```
STDIN                   FUNCTION
-----                   --------
abccde          →       s = "abccde"
```

Sample Output

```
abcdab
```

Explanation

Some of the special strings that are lexicographically greater than s are "abcdde", "abcdab", "abcdbc"

▶ Sample Case 1

STDIN		FUNCTION
abccde	→	s = "abccde"

Sample Output

abcdab

Explanation

Some of the special strings that are lexicographically greater than *s* are "abcdde", "abcdab", "abcdbc"

▼ Sample Case 1

Sample Input For Custom Testing

STDIN		FUNCTION
zzab	→	s = "zzab"

Sample Output

```
-1
```

Explanation

There is no special string of length 4 that is lexicographically greater than *s*.

```
mport java.util.concurrent.*;
mport java.util.function.*;
mport java.util.regex.*;
mport java.util.stream.*;
mport static java.util.stream.Collectors.joining;
mport static java.util.stream.Collectors.toList;
```

```
class Result {
        I
    /*
     * Complete the 'getSpecialString' function below.
     *
     * The function is expected to return a STRING.
     * The function accepts STRING s as parameter.
     */

    public static String getSpecialString(String s) {
    // Write your code here

    }

}

> public class Solution {⋯
```