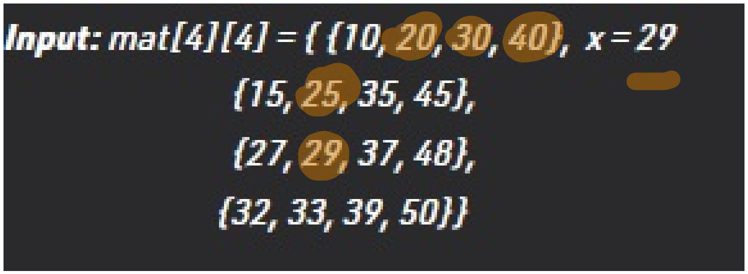
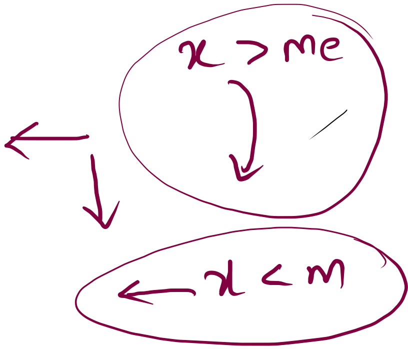
Search in sorted 2D matrix.



```
Input: mat[4][4] = { {10, 20, 30, 40},  x = 29
                     {15, 25, 35, 45},
                     {27, 29, 37, 48},
                     {32, 33, 39, 50}}
```



2, 1

2
1

Target String

Take Two Strings as input. First string as **"str"** and second string as a **"Target"** string.

You are allowed to **rotate** the original string "str" **multiple** times.

Print **"True"** if **"Target"** string can be achieved by rotating the **"str"** any number of times else print **"False"**.

Note: String **"bcda"** is a rotation of **"abcd"** but **"bdca"** is not a rotation of String **"abcd"**.

Sample Input 0

```
abcde
cdeab
```

Sample Output 0

```
True
```

abcde → eabcd

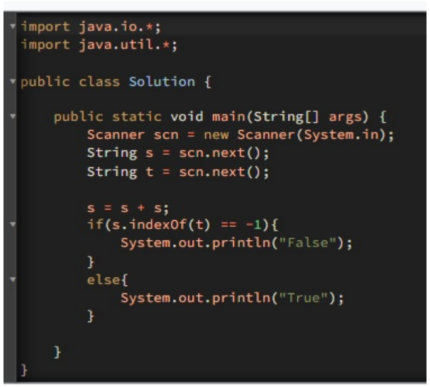
abcde. → eabcd → deabc → cdeab

✓ ab cde ab cde

s → abcde

s = s + s → abcde abcde.

pqr → -1



```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        String t = scn.next();

        s = s + s;
        if(s.indexOf(t) == -1){
            System.out.println("False");
        }
        else{
            System.out.println("True");
        }

    }
}
```

Generate Rotation

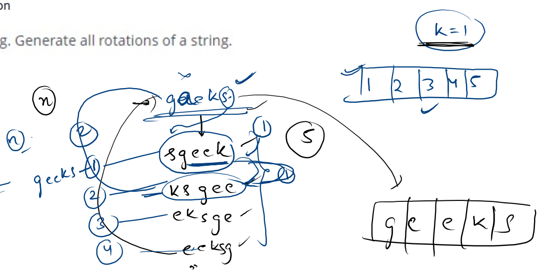
Given a string. Generate all rotations of a string.

Sample Input 0

```
geeks
```

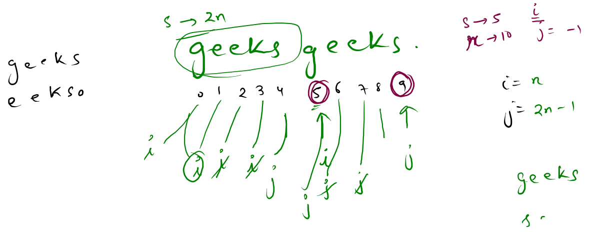
Sample Output 0

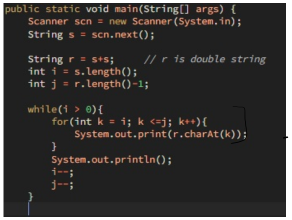
```
geeks
sgeek
ksgee
eksge
eeksg
```



A → geeks
B → sgeek
C → ksgee
D → eksge

B — 1 → C
A — 2 → C
A — 3 → D
C → 1 → D





```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String s = scn.next();

    String r = s+s;     // r is double string
    int i = s.length();
    int j = r.length()-1;

    while(i > 0){
        for(int k = i; k <=j; k++){
            System.out.print(r.charAt(k));
        }
        System.out.println();
        i--;
        j--;
    }
```

n → s.length()

O(n²)

length

5 = s → geeks

10 = r → geeksgeeks

0 1 2 3 4 5 6 7 8 9

i j

0>0

geeks
sgeek
ksgee
eksge eeksg.

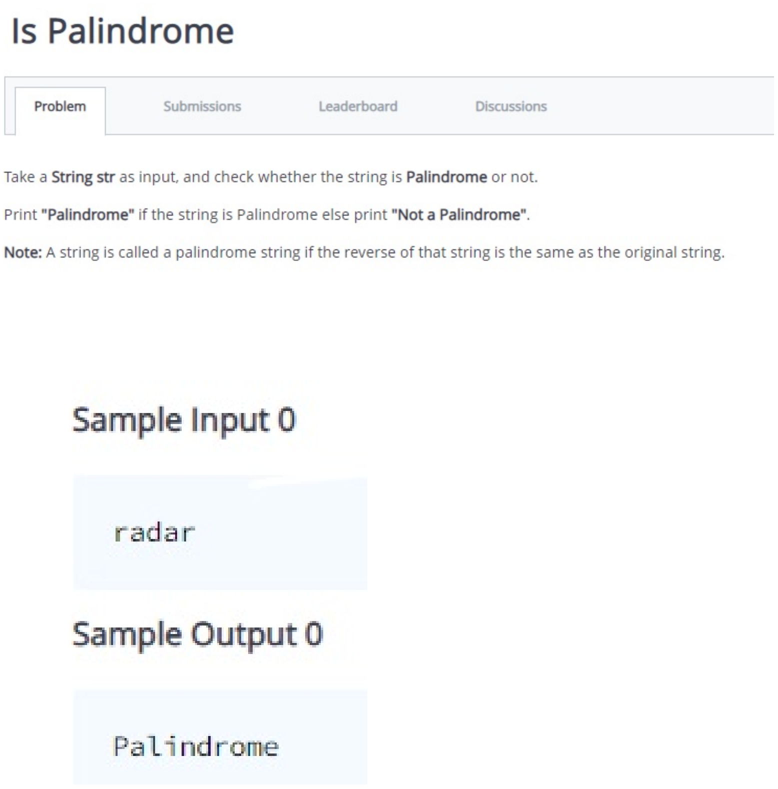
i = ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

j = ~~9~~ ~~8~~ ~~7~~ ~~6~~ ~~5~~ 4

i>0   5>0

4>0

geeks



Is Palindrome

Take a **String str** as input, and check whether the string is **Palindrome** or not.

Print **"Palindrome"** if the string is Palindrome else print **"Not a Palindrome"**.

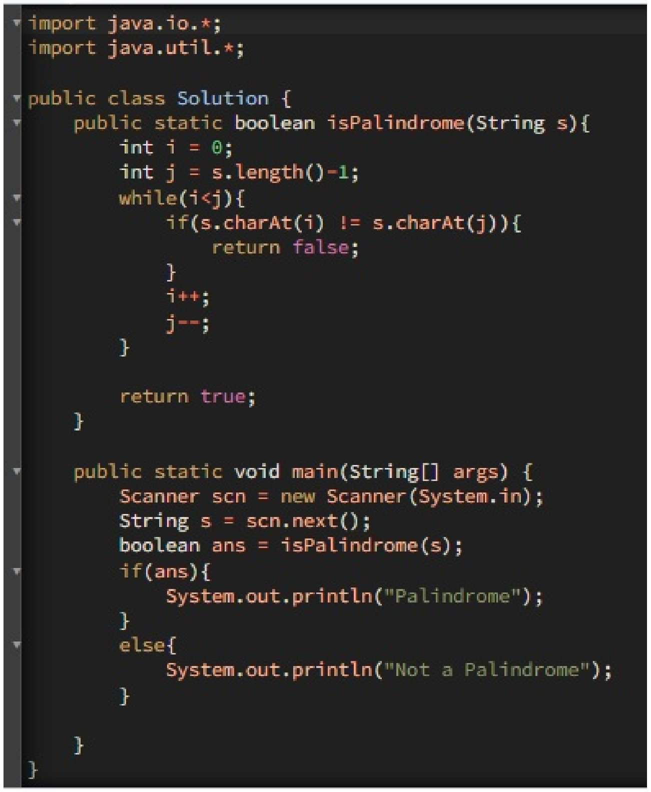
Note: A string is called a palindrome string if the reverse of that string is the same as the original string.

Sample Input 0

```
radar
```

Sample Output 0

```
Palindrome
```



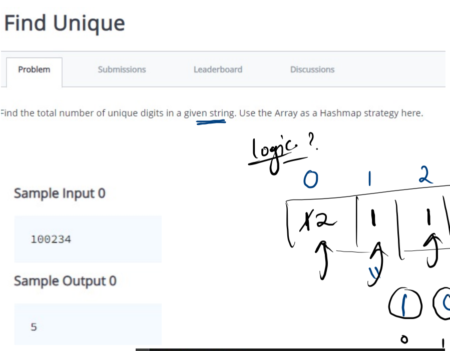
```java
import java.io.*;
import java.util.*;

public class Solution {
    public static boolean isPalindrome(String s){
        int i = 0;
        int j = s.length()-1;
        while(i<j){
            if(s.charAt(i) != s.charAt(j)){
                return false;
            }
            i++;
            j--;
        }

        return true;
    }

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        boolean ans = isPalindrome(s);
        if(ans){
            System.out.println("Palindrome");
        }
        else{
            System.out.println("Not a Palindrome");
        }

    }
}
```



Find Unique

Problem | Submissions | Leaderboard | Discussions

Find the total number of unique digits in a given string. Use the Array as a Hashmap strategy here.

Sample Input 0

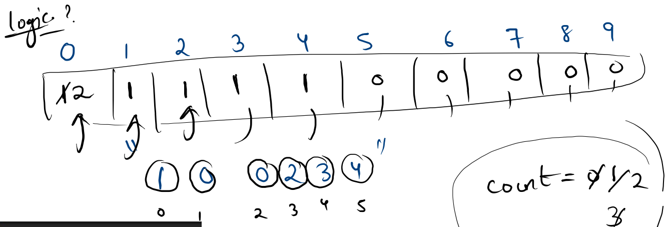
```
100234
```

Sample Output 0

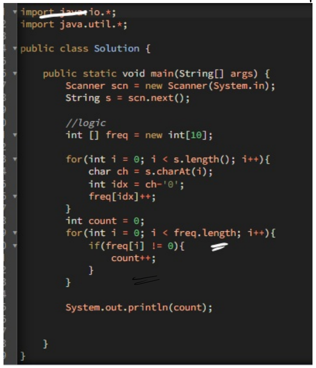
```
5
```

?!!

Logic ?



count = ~~0~~ ~~1~~ 2
~~3~~
~~4~~
5



```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();

        //logic
        int [] freq = new int[10];

        for(int i = 0; i < s.length(); i++){
            char ch = s.charAt(i);
            int idx = ch-'0';
            freq[idx]++;
        }
        int count = 0;
        for(int i = 0; i < freq.length; i++){
            if(freq[i] != 0){
                count++;
            }
        }

        System.out.println(count);

    }
}
```

1 0 0 2 3 4
0

ch = s.charAt (0) int

ch = '1' → 1

[1]

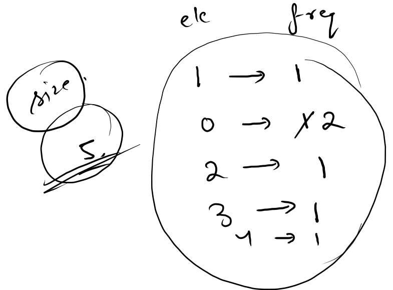
'1' - '0' → 49 - 48 = 1

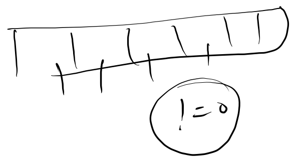
Hashmap. < K , V >

1 0 0 2 3 4

'2' → 2

'2' - '0'
= 50 - 48
= 2





Locate the Target String

Given two strings str & target, return the index where target string occurs for the first time in String str.

Sample Input 0

```
geekster
st
```

Sample Output 0

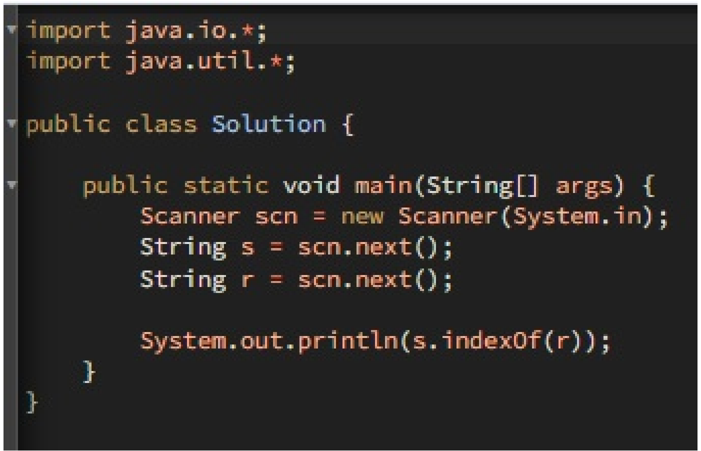
```
4
```

$s \rightarrow$ geekster st

0 1 2 3 4 5 6 7 8 9

$r \rightarrow$ st ↑

s.indexOf(r)

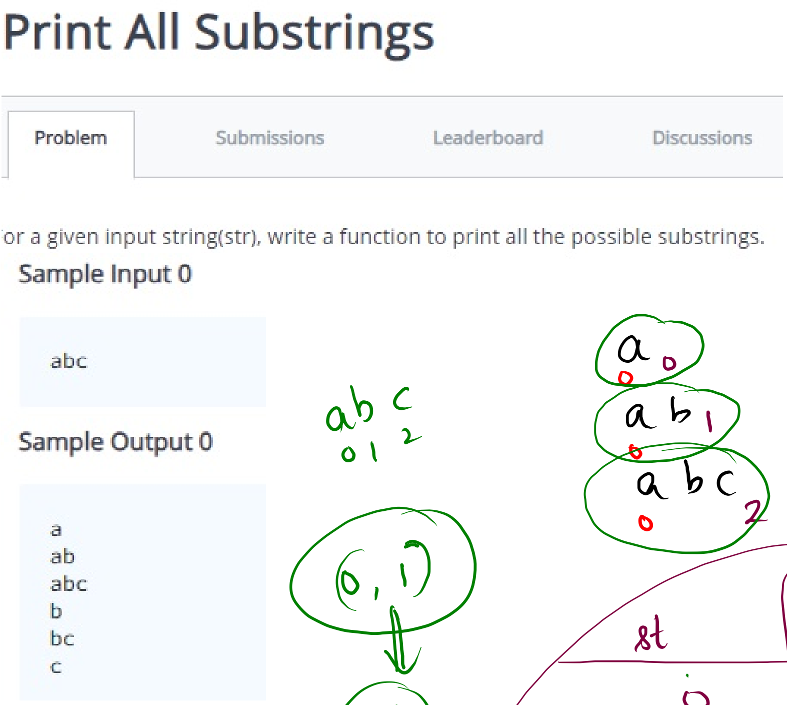


```
import java.io.*;
import java.util.*;

public class Solution {

    public static void main(String[] args) {
        Scanner scn = new Scanner(System.in);
        String s = scn.next();
        String r = scn.next();

        System.out.println(s.indexOf(r));
    }
}
```



Print All Substrings

Problem Submissions Leaderboard Discussions

or a given input string(str), write a function to print all the possible substrings.

Sample Input 0

```
abc
```

Sample Output 0

```
a
ab
abc
b
bc
c
```

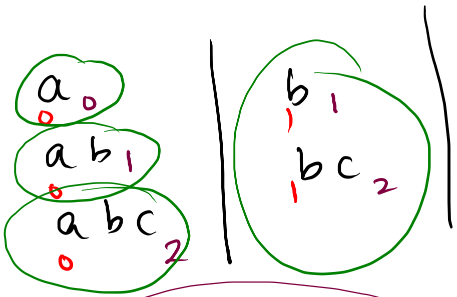
$\longrightarrow$
a b c
0 1 2

$\longrightarrow$
1. L → R
2. continuous.

ab c
0 1 2

(0, 1) ⇓ ab

s.substring(0,1)
[0,1)



st	end
0	0, 1, 2
1	1, 2
2	2

st	end
1	2

[0,1]