Lecture 3 - Strings

Correct Statement

Send Feedback

Select the correct statement(s) about strings.

Options

Attempts left: 1/2

This problem may have one or more correct answers

- [x] String is a non-primitive datatype. ✔
- [x] The length() function returns an integer value. ✔
- [x] The maximum length of String in java is 2,14,74,83,647. ✔
- [x] Strings can store spaces as well. ✔

Hurray! Correct Answer

Solution Description

The String class length method returns an int, the maximum length that would be returned by the method would be Integer.MAX_VALUE, which is 2^31 - 1, which is equivalent to 2,14,74,83,647.

Return type

Send Feedback

What is the return type of charAt() function in String class?

Options

This problem has only one correct answer

- ( ) int
- (•) char
- ( ) float
- ( ) void

Hurray! Correct Answer

Predict the output

Send Feedback

What will be the output of the following code:

```
String a ="abcd";
String b="abcda";
System.out.println(a.compareTo(b));
```

Options

Attempts left: 1/2

This problem has only one correct answer

- ( ) 0
- ( ) 1
- (•) -1
- ( ) error

Hurray! Correct Answer

Solution Description

The compareTo function returns the difference of length of strings when they are not equal.In case if the lengths are equal, then it returns the non-zero difference in ASCII values starting from 0th index to (n-1)st index and if all the indices of string are same, it returns 0.

Predict the output

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String a="coding",b="ninjas";
if(a.contains("ing"))
{
   a+=b;
}
else
{
   b+="ing";
}
System.out.print(b+a);
}
```

Answer

ninjascodingninjas ✔

Correct Answer

Predict the output

Send Feedback

What will be the output of the following code:

```
String a="coding";
for(int i=2;i<4;i++)
{
   System.out.print(a.substring(i));
}
```

Answer

dinging ✔

Correct Answer

Solution Description

For i=2, the substring will be ding and for i=3 , it will be ing

Predict the output

Send Feedback

What will be the output of the following code:

```
String a="coding";
for(int i=2;i<5;i++)
{
   System.out.print(a.substring(i-2,i+1));
}
```

Answer

cododidin ✔

Correct Answer

Correct Statement

Send Feedback

Select the correct statement.

Options

Attempts left: 1/2

This problem may have one or more correct answers

- [x] System.in in Scanner represents input Stream. ✔
- [x] If we want to read from a file then we can pass filename instead of System.in ✔
- [ ] .next() reads till a '/n' is encountered.
- [x] .nextLine() reads till '/n' is encountered. ✔

Hurray! Correct Answer

Predict the output

Send Feedback

What will be the output of the following code if the input is "java is an object-oriented language":

```
public static void main (String[] args) {
Scanner s=new Scanner(System.in);
String str1=s.next();
String str2=s.nextLine();
String str3=str2+str1;
System.out.println(str3);
}
```

Options

This problem has only one correct answer

- ( ) javais an object-oriented language
- ( ) is an object-oriented language java
- ( ) java is an object oriented language
- (•) is an object-oriented languagejava

Hurray! Correct Answer

Count Words

Send Feedback

For a given input string(str), find and return the total number of words present in it.

It is assumed that two words will have only a single space in between. Also, there wouldn't 'be any leading and trailing spaces in the given input string.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The only line of output prints an integer value denoting the tool number of words present in the string.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 sec

Sample Input 1:

Coding Ninjas!

Sample Output 1:

```
2
Sample Input 2:
this is a sample string
Sample Output 2:
5

public class Solution {

   public static int countWords(String str) {
       //Your code goes here
       int count = 0;
       for(int i=0; i<str.length(); i++){
           if(str.charAt(i)==' '){
               count++;
           }
       }
       if(str.length()==0)
           return 0;
       else
           return count+1;
   }

}
```

String Palindrome

Send Feedback

Given a string, determine if it is a palindrome, considering only alphanumeric characters.

Palindrome

A palindrome is a word, number, phrase, or other sequences of characters which read the same backwards and forwards.

Example:

If the input string happens to be, "malayalam" then as we see that this word can be read the same as forward and backwards, it is said to be a valid palindrome.

The expected output for this example will print, 'true'.

From that being said, you are required to return a boolean value from the function that has been asked to implement.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false'.

Note:

You are not required to print anything. It has already been taken care of.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1 :

```
abcdcba
Sample Output 1 :
true
Sample Input 2:
coding
Sample Output 2:
false

public class Solution {

   public static boolean isPalindrome(String str) {
       //Your code goes here

       int j=0;
       boolean flag = true;

       for(int i=0; i<str.length(); i++){
           j=str.length()-1-i;
           if(i>=j){
               break;
           }
           else if(str.charAt(i)==str.charAt(j)){
               continue;
           }
           else{
               flag = false;
               break;
           }
       }
       return flag;
   }

}
```

Predict the output

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String str1="abc";
String str2=new String("abc");
System.out.println(str1==str2);
}
```

Options

This problem has only one correct answer

- ○ true
- ◉ false

Hurray! Correct Answer

Predict the output

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String str1="abc";
String str2=new String("abc");
System.out.println(str1.equals(str2));
}
```

Options

This problem has only one correct answer

- ◉ true
- ○ false

Hurray! Correct Answer

Output problem

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String str1="abc";
String str2="bc";
String str3="ab"+str2;
System.out.println(str1==str3);
}
```

Options

This problem has only one correct answer

- ○ true
- ◉ false

Hurray! Correct Answer

Output problem

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String str1="abc";
String str2=str1+" ";
System.out.println(str1==str2);
}
```

Options

This problem has only one correct answer

- ○ true
- ◉ false

Hurray! Correct Answer

Predict the output

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
String str1="abc";
String str2="";
System.out.println(str1.contains(str2));
}
```

Options

This problem has only one correct answer

- ◉ true
- ○ false

Hurray! Correct Answer

All substrings

Send Feedback

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

Substring

A substring is a contiguous sequence of characters within a string.

Example: "cod" is a substring of "coding". Whereas, "cdng" is not as the characters taken are not contiguous

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. All the characters in the string would be in lower case.

Output Format:

Print the total number of substrings possible, where every substring is printed on a single line and hence the total number of output lines will be equal to the total number of substrings.

Note:

The order in which the substrings are printed, does not matter.

Constraints:

0 <= *N* <= 10^6

Where N is the length of the input string.

Time Limit: 1 second

Sample Input 1:

abc

Sample Output 1:

```
a
ab
abc
b
bc
c
Sample Input 2:
co
Sample Output 2:
c
co
o

public class Solution {

   public static void printSubstrings(String str) {
       //Your code goes here


       for(int i=0; i<str.length(); i++){
           int start = i;
           for(int j=i; j<str.length(); j++){
               System.out.println(str.substring(start, j+1));
           }
       }
   }

}
```

Stringbuffer

Send Feedback

Would the Stringbuffer store its string in string pool?

Options

This problem has only one correct answer

- Yes
- No

Hurray! Correct Answer

Predict the output

Send Feedback

What will be the output of the following code:

```
public static void main (String[] args) {
StringBuffer str1=new StringBuffer("");
for(int i=0;i<5;i++)
{
   str1.append((char)('a'+i));
}
System.out.println(str1);
}
```

Options

This problem has only one correct answer

- 979899100101
- abcde
- no output
- error

Hurray! Correct Answer

Reverse String Wordwise

Send Feedback

Reverse the given string word wise. That is, the last word in given string should come at 1st place, last second word at 2nd place and so on. Individual words should remain as it is.

Input format :

String in a single line

Output format :

Word wise reversed string in a single line

Constraints :

0 <= |S| <= 10^7

where |S| represents the length of string, S.

Sample Input 1:

Welcome to **Coding Ninjas**

Sample Output 1:

Ninjas Coding to **Welcome**

Sample Input 2:

Always indent your code

Sample Output 2:

code your indent **Always**

```
public class Solution {
   public static String reverseWordWise(String input) {
       // Write your code here
```

```
        String ans = "";
        int end = input.length();

        for(int i=input.length()-1; i>0; i--){
            if(input.charAt(i)==' '){
                ans += input.substring(i+1, end);
                ans += " ";
                end = i;
            }
        }
        for(int i=0; i<input.length(); i++)
            if(input.charAt(i)==' '){
                ans += input.substring(0,i);
                break;
            }
        return ans;

    }
}
```

Check Permutation

Send Feedback

For a given two strings, 'str1' and 'str2', check whether they are a permutation of each other or not.

Permutations of each other

Two strings are said to be a permutation of each other when either of the string's characters can be rearranged so that it becomes identical to the other one.

Example:
str1= "sinrtg"
str2 = "string"

The character of the first string(str1) can be rearranged to form str2 and hence we can say that the given strings are a permutation of each other.

Input Format:

The first line of input contains a string without any leading and trailing spaces, representing the first string 'str1'.

The second line of input contains a string without any leading and trailing spaces, representing the second string 'str2'.

Note:

All the characters in the input strings would be in lower case.

Output Format:

The only line of output prints either 'true' or 'false', denoting whether the two strings are a permutation of each other or not.

You are not required to print anything. It has already been taken care of. Just implement the function.

Constraints:

0 <= N <= 10^6

Where N is the length of the input string.

```
Time Limit: 1 second
Sample Input 1:
abcde
baedc
Sample Output 1:
true
Sample Input 2:
abc
cbd
Sample Output 2:
false

public class Solution {

   public static boolean isPermutation(String str1, String str2) {
       //Your code goes here
       int[] arr = new int[26];
       boolean flag = true;
       for(int i=0; i<str1.length(); i++){
           arr[str1.charAt(i)-97]++;
       }
       for(int i=0; i<str2.length(); i++){
           arr[str2.charAt(i)-97]--;
       }
       for(int i=0; i<26; i++){
           if(arr[i]!=0){
               flag = false;
               break;
           }
       }
       return flag;
   }

}
```

Remove_Consecutive_Duplicates

```
Send Feedback
For a given string(str), remove all the consecutive duplicate characters.
Example:
Input String: "aaaa"
Expected Output: "a"

Input String: "aabbbcc"
Expected Output: "abc"
Input Format:
The first and only line of input contains a string without any leading and
trailing spaces. All the characters in the string would be in lower case.
Output Format:
The only line of output prints the updated string.
Note:
You are not required to print anything. It has already been taken care of.
Constraints:
```

```
0 <= N <= 10^6
Where N is the length of the input string.

Time Limit: 1 second
Sample Input 1:
aabccbaa
Sample Output 1:
abcba
Sample Input 2:
xxyyzxx
Sample Output 2:
xyzx


public class Solution {

   public static String removeConsecutiveDuplicates(String str) {
       //Your code goes here
       String str1 = "";

       for(int i=0; i<str.length(); i++){
           if(i>0 && str.charAt(i-1) == str.charAt(i))
               continue;
           str1 += str.charAt(i);
       }
       return str1;
   }
}
```

Reverse Each Word

Send Feedback

Aadil has been provided with a sentence in the form of a string as a function parameter. The task is to implement a function so as to print the sentence such that each word in the sentence is reversed.

Example:

Input Sentence: "Hello, I am Aadil!"

The expected output will print, ",olleH I ma !lidaA".

Input Format:

The first and only line of input contains a string without any leading and trailing spaces. The input string represents the sentence given to Aadil.

Output Format:

The only line of output prints the sentence(string) such that each word in the sentence is reversed.

Constraints:

```
0 <= N <= 10^6
Where N is the length of the input string.

Time Limit: 1 second
Sample Input 1:
Welcome to Coding Ninjas
Sample Output 1:
emocleW ot gnidoC sajniN
```

```
Sample Input 2:
Always indent your code
Sample Output 2:
syawlA tnedni ruoy edoc


public class Solution {

   public static String reverseEachWord(String str) {
       //Your code goes here

       int start = 0;
       String ans = "";

       for(int i=0; i<str.length(); i++){
           String reversedWord = "";
           if(str.charAt(i)==' '){
               for(int j=start; j<i; j++){
                   reversedWord = str.charAt(j) + reversedWord;
               }
               ans += reversedWord;
               ans += " ";
               start = i+1;
           }
       }
       String reversedWord = "";
       for(int i=start; i<str.length(); i++){
           reversedWord = str.charAt(i) + reversedWord;
       }
       ans += reversedWord;
       return ans;
   }
}
```

Remove character

Send Feedback

For a given a string(str) and a character X, write a function to remove all the occurrences of X from the given string.

The input string will remain unchanged if the given character(X) doesn't' exist in the input string.

Input Format:

The first line of input contains a string without any leading and trailing spaces.

The second line of input contains a character(X) without any leading and trailing spaces.

Output Format:

The only line of output prints the updated string.

Note:

You are not required to print anything explicitly. It has already been taken care of.

Constraints:

```
0 <= N <= 10^6
Where N is the length of the input string.

Time Limit: 1 second
Sample Input 1:
aabccbaa
a
Sample Output 1:
bccb
Sample Input 2:
xxyyzxx
y
Sample Output 2:
xxzxx


public class Solution {

   public static String removeAllOccurrencesOfChar(String str, char ch) {
       // Your code goes here
       String ans = "";

       for(int i=0; i<str.length(); i++){
           if(str.charAt(i) == ch)
               continue;
           ans += str.charAt(i);
       }
       return ans;
   }

}
```

```
Highest Occuring Character
Send Feedback
For a given a string(str), find and return the highest occurring
character.
Example:
Input String: "abcdeapapqarr"
Expected Output: 'a'
Since 'a' has appeared four times in the string which happens to be the
highest frequency character, the answer would be 'a'.
If there are two characters in the input string with the same frequency,
return the character which comes first.
Consider:
Assume all the characters in the given string to be in lowercase always.
Input Format:
The first and only line of input contains a string without any leading and
trailing spaces.
Output Format:
The only line of output prints the updated string.
Note:
You are not required to print anything explicitly. It has already been
taken care of.
Constraints:
```

```
0 <= N <= 10^6
Where N is the length of the input string.

Time Limit: 1 second
Sample Input 1:
abdefgbabfba
Sample Output 1:
b
Sample Input 2:
xy
Sample Output 2:
x


public class Solution {

   public static char highestOccuringChar(String str) {
       //Your code goes here
       int[] frequency = new int[26];
       char ans;

       for(int i=0; i<str.length(); i++){
           frequency[str.charAt(i)-97]++;
       }
       int max = 0;
       for(int i=0; i<26; i++){
           if(frequency[max]<frequency[i])
               max = i;
       }
       ans = (char)(max+97);
       return ans;
   }
}
```

Compress the String

Send Feedback

Write a program to do basic string compression. For a character which is consecutively repeated more than once, replace consecutive duplicate occurrences with the count of repetitions.

Example:

If a string has 'x' repeated 5 times, replace this "xxxxx" with "x5".

The string is compressed only when the repeated character count is more than 1.

Note:

Consecutive count of every character in the input string is less than or equal to 9.

Input Format:

The first and only line of input contains a string without any leading and trailing spaces.

Output Format:

The output contains the string after compression printed in single line.

Note:

You are not required to print anything. It has already been taken care of. Just implement the given function.
Constraints:
0 <= N <= 10^6

Where 'N' is the length of the input string.

Time Limit: 1 sec
Sample Input 1:
aaabbccdsa
Sample Output 1:
a3b2c2dsa
Explanation for Sample Output 1:
In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is repeated 2 times and 'd', 's' and 'a' and occuring 1 time hence no compression for last 3 characters.
Sample Input 2:
aaabbcddeeeee
Sample Output 2:
a3b2cd2e5
Explanation for Sample Output 2:
In the given string 'a' is repeated 3 times, 'b' is repeated 2 times, 'c' is occuring single time, 'd' is repeating 2 times and 'e' is repeating 5times.

```java
public class Solution {
   public static String getCompressedString(String str) {
       //Your code goes here
       String str1 = "";
       int count = 0;
       for(int i=0; i<str.length(); i++){
           if(i>0 && str.charAt(i-1) == str.charAt(i)){
               count++;
               continue;
           }
           else if(i>0 && count>0){
               str1 += count+1;
               count = 0;
           }
           str1 += str.charAt(i);
       }
       if(count>0)
           str1 += count+1;
       return str1;
   }

}
```