**Strings Method Examples**

**1.Program to display string value**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Print the string and goto step 4

step 4: STOP

Program

**import** java.util.Scanner;  
  
**public class** StringDisplay {  
  
**public static void** main(String[] args) {  
 String st = **new** String();  
 Scanner sc = **new** Scanner(System.***in***);  
 System.***out***.println(**"Enter the string"**);  
 st = sc.nextLine();  
 System.***out***.println(st);  
 }  
}

I/P

Enter the string

Welcome

O/P

welcome

**2.Program to implement equals method**

Algorithm

step 1:START

step 2: Read two string values and goto step 3

step 3: Check s2.equals(s1) and store result in b1.Goto step 4

step 4: Check if condition.If true,goto step 5 else goto step 6

step 5: Print statement and goto step 7

step 6: Print statement and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** EqualsMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
**boolean** b1 = s2.equals(s1);  
**if** (b1)  
 System.***out***.println(**"The strings are equal"**);  
**else**System.***out***.println(**"The strings are not equal"**);  
  
 }  
}

I/P

Enter first string

hi

Enter second string

Hi

O/P:

The strings are equal

**3.Program to implement contentEquals method**

Algorithm

step 1:START

step 2: Read two string values and goto step 3

step 3: Check s2.contentEquals(s1) and store result in b1.Goto step 4

step 4: Check if condition.If true,goto step 5 else goto step 6

step 5: Print statement and goto step 7

step 6: Print statement and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** ContentEqualsMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
**boolean** b1 = s2.contentEquals(s1);  
**if** (b1)  
 System.***out***.println(**"The strings are equal"**);  
**else**System.***out***.println(**"The strings are not equal"**);  
  
 }  
}

I/P

Enter first string

hello

Enter second string

Hell

O/P

The strings are not equal

**4.Program to implement equalsIgnoreCase method**

Algorithm

step 1:START

step 2: Read two string values and goto step 3

step 3: Check s2.equalsIgnoreCase(s1) and store result in b1.Goto step 4

step 4: Check if condition.If true,goto step 5 else goto step 6

step 5: Print statement and goto step 7

step 6: Print statement and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** EqualsIgnoreCaseMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
**boolean** b1 = s2.equalsIgnoreCase(s1);  
**if** (b1)  
 System.***out***.println(**"The strings are equal"**);  
**else**System.***out***.println(**"The strings are not equal"**);  
  
 }  
}

I/P

Enter first string

Hello

Enter second string

Hello

O/P:

The strings are equal

**5.Program to implement touppercase method**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Check s1.toUpperCase and store result in s2.Goto step 4

step 4: Print s1 and goto step 5

step 5: STOP

Program

**import** java.util.Scanner;  
  
**public class** ToUpperCaseMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 String s2 = s1.toUpperCase();  
 System.***out***.println(**"String in upper case is \n"**+s2);  
 }  
}

I/P

Enter string

this is statement in upper case

O/P

String in upper case is

THIS IS STATEMENT IN UPPER CASE

**6.Program to implement tolowercase method**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Check s1.toLowerCase and store result in s2.Goto step 4

step 4: Print s1 and goto step 5

step 5: STOP

Program

**import** java.util.Scanner;  
  
**public class** ToLowerCaseMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 String s2 = s1.toLowerCase();  
 System.***out***.println(**"String in lower case is \n"**+s2);  
 }  
}

I/P

Enter string

this is LOWER CASE STATement

O/P

String in lower case is

this is lower case statement

**7.Program to implement contains method**

Algorithm

step 1:START

step 2: Read two string values and goto step 3

step 3: Check s2.contains(s1) and store result in b1.Goto step 4

step 4: Check if condition.If true,goto step 5 else goto step 6

step 5: Print statement and goto step 7

step 6: Print statement and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** ContainsMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
**boolean** b1 = s2.contains(s1);  
**if** (b1)  
 System.***out***.println(**"The string \t"**+s2+**" contains \t"**+s1);  
**else**System.***out***.println(**"The string \t"**+s2+**" doesn't contain \t"**+s1);  
 }  
}

I/P

Enter first string

World

Enter second string

Welcome to My World

O/P

The string Welcome to My World contains World

**8.Program to implement indexof method**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Print the s1.indexOf value and goto step 4

step 4: STOP

Program

**public class** IndexOfMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 System.***out***.println(**"The index of character e is "**+s1.indexOf(**"e"**));  
 }  
}

I/P

Enter string

welcome

O/P

The index of character e is at 1

**9.Program to implement charat method**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Print the length of string and goto step 4

step 4: Read position value and goto step 5

step 5: Print the character at the position and goto step 6

step 6: STOP

Program

**import** java.util.Scanner;  
  
**public class** CharAtMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
**int** position;  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 System.***out***.println(**"The length of string is "**+s1.length());  
 System.***out***.println(**"Enter position to find character at that position"**);  
 position = sc.nextInt();  
 System.***out***.println(**"The character at index "**+position+**" is "**+s1.charAt(position));  
 }  
}

I/P

Enter string

This is new program

The length of string is 19

Enter position to find character at that position

18

O/P

The character at index 18 is m

**10.Program to implement toString method**

Algorithm

step 1:START

step 2: Read character array values and goto step 3

step 3: Check for loop condition.If true,goto step 4 else goto step 5

step 4: Print characters of c array and repeat step 3

step 5: Convert c to string and store it to st variable.Goto step 6

step 6: Print the value of st and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** ToStringMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
**char** c[] = {**'a'**,**'b'**,**'h'**,**'t'**};  
**for** (**int** i=0;i<c.**length**;i++){  
 System.***out***.println(**"Characters are:"**+ c[i]);  
 }  
 String st = c.toString();  
 System.***out***.println(st); *//toString() Returns the value of a String object*}  
}

I/P

Characters are:a

Characters are:b

Characters are:h

Characters are:t

O/P

[C@234bef66

**11.Program to implement compareTo method**

Algorithm

step 1:START

step 2: Read two strings and goto step 3

step 3: Assign the result of s1.compareTo(s2) to s3 and goto step 4

step 4: Print the result and goto step 5

step 5: STOP

Program

**import** java.util.Scanner;  
  
**public class** CompareToMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
**int** s3 = s1.compareTo(s2);  
 System.***out***.println(s1.compareTo(s2));  
 }  
}

I/P

Enter first string

welcome

Enter second string

hi

O/P

15

**12.Program to implement substring method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Assign the result of s1.substring(4) to b1 and goto step 4

step 4: Print the result and goto step 5

step 5: STOP

Program

**import** java.util.Scanner;  
  
**public class** SubstringMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 String b1 = s1.substring(4);  
 System.***out***.println(b1);  
 }  
}

I/P

Enter string

welcome

O/P

ome

**13.Program to implement trim method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Print the result of s1.trim and goto step 4

step 4: STOP

Program

**import** java.util.Scanner;  
  
**public class** TrimMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 System.***out***.println(s1.trim());  
 }  
}

I/P

Enter string

hi welcome

O/P

hi welcome

**14.Program to implement endswith method**

Algorithm

step 1:START

step 2: Read two strings value and goto step 3

step 3: Assign the value of s1.endsWith(s2) to b and goto step 4

step 4: Check if statement condition.If true,goto step 5 else goto step 6

step 5: Print the statement and goto step 7

step 6: Print the statement and goto step 7

step 7: STOP

Program

**import** java.util.Scanner;  
  
**public class** EndsWithMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter the string which we want to check"**);  
 s2 = sc.nextLine();  
**boolean** b = s1.endsWith(s2);  
**if** (b)  
 System.***out***.println(**"The string "**+s1+**" ends with "**+s2);  
**else**System.***out***.println(**"The string "**+s1+**" doesn't end with "**+s2);  
 }  
}

I/P

Enter string

Welcome World

Enter the string which we want to check

World

O/P

The string Welcome World ends with World

**15.Program to implement startswith method**

Algorithm

step 1:START

step 2: Read two strings value and goto step 3

step 3: Assign the value of s1.startsWith(s2) to b and goto step 4

step 4: Check if statement condition.If true,goto step 5 else goto step 6

step 5: Print the statement and goto step 7

step 6: Print the statement and goto step 7

step 7: STOP

Program:

**import** java.util.Scanner;  
  
**public class** StartsWithMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter the string which we want to check"**);  
 s2 = sc.nextLine();  
**boolean** b = s1.startsWith(s2);  
**if** (b)  
 System.***out***.println(**"The string "**+s1+**" start with "**+s2);  
**else**System.***out***.println(**"The string "**+s1+**" doesn't start with "**+s2);  
 }  
}

I/P

Enter string

Welcome World

Enter the string which we want to check

Welcome

O/P

The string Welcome World starts with Welcome

**16.Program to implement getChars method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Do s1.getChars(1,3,c,1) and goto step 4

step 4: Check for loop condition.If true,goto step 5 else goto step 6

step 5: Print c[i] values and repeat step 4

step 6: STOP

Program

**import** java.util.Scanner;  
  
**public class** GetCharMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
**char** c[] = **new char**[5];  
 String s1 = **"Welcome to new program"**;  
 s1.getChars(1,3,c,1);  
**for** (**int** i=0;i<c.**length**;i++){  
 System.***out***.println(c[i]);  
 }  
 }  
}

I/P

Enter string

hello world

O/P

e

l

**17.Program to implement getbytes method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Do s1.getBytes() and assign it to array1 and goto step 4

step 4: Check for loop condition.If true,goto step 5 else goto step 6

step 5: Print b values and repeat step 4

step 6: STOP

Program

**import** java.util.Scanner;  
  
**public class** GetBytesMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
**byte**[] array1 = s1.getBytes();  
 System.***out***.print(**"Default Charset encoding:"**);  
**for**(**byte** b: array1){  
 System.***out***.print(b);  
 }  
 }  
}

I/P

Enter string

welcome

O/P

Default Charset encoding:11910110899111109101

**18.Program to implement intern method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3:Print s1.intern value and goto step 4

step 4: STOP

Program

**import** java.util.Scanner;  
  
**public class** InternMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter string"**);  
 s1 = sc.nextLine();  
 System.***out***.println(s1.intern());  
 }  
}

I/P

Enter string

welcome

O/P

welcome

**19.Program to implement valueof method**

Algorithm

step 1:START

step 2: Read character array value and goto step 3

step 3: Convert vowel to string and goto step 4

step 4: Print the string and goto step 5

step 5: STOP

Program

**public class** ValueOfMethod {  
**public static void** main(String args[]){  
**char** vowel[] = {**'A'**, **'E'**, **'I'**, **'O'**, **'U'**};  
 String str = String.*valueOf*(vowel);*// Converting array to string*System.***out***.println(str);  
 }  
}

O/P

AEIOU

**20.Program to implement format method**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Assign string.format to formattedString and goto step 4

step 4: Assign string.format to formattedString2 and goto step 5

step 5: Print formattedString and formattedString2 and goto step 6

step 6: STOP

Program

**public class** FormatMethod {  
**public static void** main(String args[]){  
 String str = **"just a string"**;  
  
*//concatenating string using format*String formattedString = String.*format*(**"My String is %s"**, str);  
  
*/\*formatting the value passed and concatenating at the same time  
 \* %.6f is for having 6 digits in the fractional part  
 \*/*String formattedString2 = String.*format*(**"My String is %.6f"**,12.121);  
  
 System.***out***.println(formattedString);  
 System.***out***.println(formattedString2);  
 }  
}

O/P:

My String is just a string

My String is 12.121000

**21.Program to implement split method**

Algorithm

Program

**public class** SplitMethod {  
**public static void** main(String args[]){  
*// This is out input String*String str = **new** String(**"28/12/2013"**);  
  
 System.***out***.println(**"split(String regex):"**);  
*/\* Here we are using first variation of java string split method  
 \* which splits the string into substring based on the regular  
 \* expression, there is no limit on the substrings  
 \*/*String array1[]= str.split(**"/"**);  
**for** (String temp: array1){  
 System.***out***.println(temp);  
 }  
  
*/\* Using second variation of split method here. Since the limit is passed  
 \* as 2. This method would only produce two substrings.  
 \*/*System.***out***.println(**"split(String regex, int limit) with limit=2:"**);  
 String array2[]= str.split(**"/"**, 2);  
**for** (String temp: array2){  
 System.***out***.println(temp);  
 }  
  
 System.***out***.println(**"split(String regex, int limit) with limit=0:"**);  
 String array3[]= str.split(**"/"**, 0);  
**for** (String temp: array3){  
 System.***out***.println(temp);  
 }  
  
*/\* When we pass limit as negative. The split method works same as the first variation  
 \* because negative limit says that the method returns substrings with no limit.  
 \*/*System.***out***.println(**"split(String regex, int limit) with limit=-5:"**);  
 String array4[]= str.split(**"/"**, -5);  
**for** (String temp: array4){  
 System.***out***.println(temp);  
 }  
 }  
}

O/P:

split(String regex):

28

12

2013

split(String regex, int limit) with limit=2:

28

12/2013

split(String regex, int limit) with limit=0:

28

12

2013

split(String regex, int limit) with limit=-5:

28

12

2013

**22.Program to implement concat method**

Algorithm

step 1:START

step 2: Read two string values and goto step 3

step 3: Print the s1.concat value and goto step 4

step 4: STOP

Program

**import** java.util.Scanner;  
  
**public class** ConcatMethod {  
**public static void** main(String args[]) {  
 Scanner sc = **new** Scanner(System.***in***);  
 String s1 = **new** String();  
 System.***out***.println(**"Enter first string"**);  
 s1 = sc.nextLine();  
 String s2 = **new** String();  
 System.***out***.println(**"Enter second string"**);  
 s2 = sc.nextLine();  
 System.***out***.println(s1.concat(s2));  
 }  
}

I/P

Enter first string

hello

Enter second string

World

O/P

Helloworld

**23.Program to know linecount**

Algorithm

step 1:START

step 2: Read string and goto step 3

step 3: Call getlinecount method and store the result in count.Goto step 4

step 4: Return the value of text.split("[\n|\r]").length and goto step 5

step 5: Return from method and print the value of count and goto step 6

step 6: STOP

Program

**public class** LineCount {  
**public static int** getLineCount(String text){  
  
**return** text.split(**"[\n|\r]"**).**length**;  
 }  
  
**public static void** main(String a[]){  
  
 String str = **"line1\nline2\nline3\rline4"**;  
 System.***out***.println(str);  
**int** count = *getLineCount*(str);  
 System.***out***.println(**"line count: "**+count);  
 }  
}

O/P

line1

line2

line4

line count: 4

**24.Program to find longest substring in array of strings**

Algorithm

step 1:START

step 2: Read string and goto step 3

step 3: Call findstems method and store the result in stems and goto step 4.

step 4: Assign arr[0] to s and goto step 5

step 5: Assign s.length to len and goto step 6

step 6: Check for loop condition.If true,goto step 5 else goto step 16

step 7: Check for loop condition.If tru,goto step 8 else repeat step 6

step 8: Assign s.subsring(i,j) to stem and goto step 9

step 9: Assign k value to 1 and goto step 10

step 10: Check for loop condition.If true,goto step 11 else goto step

step 11: Check if statement condition.If tru,goto step 12 else repeat step 10

step 12: Break the loop and goto step 16

step 13: Check if statement condition.If true,goto step 14 else goto step 15

step 14: Assign stem to res and repeat step 7

step 15: Return res and goto step 16

step 16: Return from the method and print stems value.Goto step 17

step 17: STOP

Program

**public class** LongestCommonSubString {  
**public static** String findstem(String arr[])  
 {  
*// Determine size of the array***int** n = arr.**length**;  
  
*// Take first word from array as reference*String s = arr[0];  
**int** len = s.length();  
  
 String res = **""**;  
  
**for** (**int** i = 0; i < len; i++) {  
**for** (**int** j = i + 1; j <= len; j++) {  
  
*// generating all possible substrings  
 // of our reference string arr[0] i.e s*String stem = s.substring(i, j);  
**int** k = 1;  
**for** (k = 1; k < n; k++)  
  
*// Check if the generated stem is  
 // common to all words***if** (!arr[k].contains(stem))  
**break**;  
  
*// If current substring is present in  
 // all strings and its length is greater  
 // than current result***if** (k == n && res.length() < stem.length())  
 res = stem;  
 }  
 }  
  
**return** res;  
 }  
  
*// Driver Code***public static void** main(String args[])  
 {  
 String arr[] = { **"grace"**, **"graceful"**, **"disgraceful"**,  
**"gracefully"** };  
 String stems = *findstem*(arr);  
 System.***out***.println(stems);  
 }  
}

O/P

Grace

**25.Program to find no.of character corrections to make strings equal**

Algorithm

step 1:START

step 2: Read string values and goto step 3

step 3: Assign string a length to n and goto step 4

step 4: Call minOperations method and goto step 5

step 5: Check for loop condition.If true,goto step 6 else goto step 12

step 6: Assign the values of a.charAt(i) to x,b.charAt(i) to y and c.charAt(i) to z and goto step 7

step 7: Check if statement condition.If true,goto step 8 else goto step 9

step 8: No operations required and repeat step 5

step 9: Check else if condition.If true,goto step 10 else goto step 11

step 10: Increment ans value by 1 and repeat step 5

step 11: Increment ans value by 2 and repeat step 5

step 12: Return ans value and goto step 13

step 13: Return from minOperations method and print he result from the method.Goto step 14

step 14: STOP

Program

**public class** NoOfCharToEqual {  
**static int** minOperations(**int** n, String a, String b, String c)  
 {  
  
*// To store the count of operations***int** ans = 0;  
**for** (**int** i = 0; i < n; i++) {  
**char** x = a.charAt(i);  
**char** y = b.charAt(i);  
**char** z = c.charAt(i);  
  
*// No operation required***if** (x == y && y == z)  
 ;  
  
*// One operation is required when  
 // any two characters are equal***else if** (x == y || y == z || x == z) {  
 ans++;  
 }  
  
*// Two operations are required when  
 // none of the characters are equal***else** {  
 ans += 2;  
 }  
 }  
  
*// Return the minimum count of operations required***return** ans;  
 }  
  
*// Driver code***public static void** main(String[] args)  
 {  
 String a = **"place"**;  
 String b = **"abcde"**;  
 String c = **"plybe"**;  
**int** n = a.length();  
 System.***out***.print(*minOperations*(n, a, b, c));  
 }  
}

O/P

6

**26.Program to know number of substrings that contain given character k times**

Program

**public class** SubstringKTimes {  
*// Function to return the count of required sub-strings***static int** countSubString(**char**[] s, **char** c, **int** k)  
 {  
  
*// Left and right counters for characters on  
 // both sides of sub-string window***int** leftCount = 0, rightCount = 0;  
  
*// Left and right pointer on both  
 // sides of sub-string window***int** left = 0, right = 0;  
  
*// Initialize the frequency***int** freq = 0;  
  
*// Result and length of string***int** result = 0, len = s.**length**;  
  
*// Initialize the left pointer***while** (s[left] != c && left < len)  
 {  
 left++;  
 leftCount++;  
 }  
  
*// Initialize the right pointer*right = left + 1;  
**while** (freq != (k - 1) && (right - 1) < len)  
 {  
**if** (s[right] == c)  
 freq++;  
 right++;  
 }  
  
*// Traverse all the window sub-strings***while** (left < len && (right - 1) < len)  
 {  
  
*// Counting the characters on left side  
 // of the sub-string window***while** (s[left] != c && left < len)  
 {  
 left++;  
 leftCount++;  
 }  
  
*// Counting the characters on right side  
 // of the sub-string window***while** (right < len && s[right] != c)  
 {  
**if** (s[right] == c)  
 freq++;  
 right++;  
 rightCount++;  
 }  
  
*// Add the possible sub-strings  
 // on both sides to result*result = result + (leftCount + 1) \* (rightCount + 1);  
  
*// Setting the frequency for next  
 // sub-string window*freq = k - 1;  
  
*// Reset the left and right counters*leftCount = 0;  
 rightCount = 0;  
  
 left++;  
 right++;  
 }  
**return** result;  
 }  
  
*// Driver code***public static void** main(String[] args)  
 {  
 String s = **"abada"**;  
**char** c = **'a'**;  
**int** k = 2;  
  
 System.***out***.println(*countSubString*(s.toCharArray(), c, k));  
 }  
}

O/P

4

**27.Program to find minimum number of swaps to make two binary strings equal**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Call minSwaps method and goto step 4

step 4: Check for loop condition.If true,goto step 5 else goto step 9

step 5: Check if condition.If true,goto step 6 else goto step 7

step 6: Increment c0 value by 1 and repeat step 4

step 7: Increment c1 value by 1 and repeat step 4

step 8: Assign the value of c0 / 2 + c1 / 2 to ans and got step 9

step 9: Check if statement condition.If true,goto step 10 else goto step 11

step 10: Return ans value and goto step 14

step 11: Check the condition.If true,goto step 12 else goto step 13

step 12: Return ans+2 value and goto step 14

step 13: Return -1 value and goto step 14

step 14: Return from minSwaps method and assign the value to ans and goto step 15

step 15: Print ans value and goto step 16

step 16: STOP

Program

**public class** SwapsBinaryEqual {  
**static int** minSwaps(String s1, String s2)  
 {  
  
**int** c0 = 0, c1 = 0;  
  
**for** (**int** i = 0; i < s1.length(); i++)  
 {  
*// Count of zero's***if** (s1.charAt(i) == **'0'** && s2.charAt(i) == **'1'**)  
 {  
 c0++;  
 }  
  
*// Count of one's***else if** (s1.charAt(i) == **'1'** && s2.charAt(i) == **'0'**)  
 {  
 c1++;  
 }  
 }  
  
**int** ans = c0 / 2 + c1 / 2;  
  
**if** (c0 % 2 == 0 && c1 % 2 == 0)  
 {  
**return** ans;  
 }  
**else if** ((c0 + c1) % 2 == 0)  
 {  
**return** ans + 2;  
 }  
**else**{  
**return** -1;  
 }  
 }  
  
*// Driver code***public static void** main (String[] args)  
 {  
  
 String s1 = **"0011"**, s2 = **"1111"**;  
**int** ans = *minSwaps*(s1, s2);  
  
 System.***out***.println(ans);  
  
 }  
}

O/P

1

**28.Program to find minimum number of appends needed to make string a palindrome**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Call noofAppends method and goto step 4

step 4: Call ispalindrome method and goto step 5

step 5: Assign string length to len and goto step 6

step 6: Check if statement condition.If true,goto step 7 else goto step 8

step 7: Return true value and goto step 16

step 8: Assign str[0] to ptr1 and goto step 9

step 9: Assign str[len-1] to ptr2 and goto step 10

step 10: Check while loop condition.If true,goto step 11 else goto step 15

step 11: Check if statement condition.If true,goto step 12 else goto step 13

step 12: Return false and goto step 16

step 13: Increment ptr1 value by 1 and goto step 14

stp 14: Decrement ptr2 value by 1 and repeat step 10

step 15: Return true value and goto step 16

step 16: Return from ispalindrome method and goto step 17

step 17: Check if statement condition with the result from ispalindrome method.If true,goto step 18 else goto step 19

step 18: Return 0 and goto step

step 19: Assign s.substring(1) to s and goto step 20

step 20: Return the value and goto step 21

step 21: Return from noofAppends method and print the result.Goto step 22

step 22: STOP

Program

**public class** AppendsPalindrome {  
**static boolean** isPalindrome(**char** []str)  
 {  
**int** len = str.**length**;  
  
*// single character is always palindrome***if** (len == 1)  
**return true**;  
  
*// pointing to first character***char** ptr1 = str[0];  
  
*// pointing to last character***char** ptr2 = str[len-1];  
  
**while** (ptr2 > ptr1)  
 {  
**if** (ptr1 != ptr2)  
**return false**;  
 ptr1++;  
 ptr2--;  
 }  
  
**return true**;  
 }  
  
*// Recursive function to count number of appends***static int** noOfAppends(String s)  
 {  
**if** (*isPalindrome*(s.toCharArray()))  
**return** 0;  
  
*// Removing first character of string by  
 // incrementing base address pointer.*s=s.substring(1);  
  
**return** 1 + *noOfAppends*(s);  
 }  
  
*// Driver code***public static void** main(String arr[])  
 {  
 String s = **"abede"**;  
 System.***out***.printf(**"%d\n"**, *noOfAppends*(s));  
 }  
}

O/P

2

**29.Program to remove character to make a string palindrome**

Algorithm

step 1:START

step 2: Read string and goto step 3

step 3: Call possiblePalinByRemovingOneChar method and store the result in idx variable.Goto step 4

step 4: Assign the value 0 for low,string length - 1 for high and goto step 5

step 5: Check for loop condition.If true,goto step 6 else goto step

step 6: Check if statement condition.If true,goto step 7 else goto step 9

step 7: Increment low value by 1 and goto step 8

step 8: Decrement high value by 1 and repeat step 5

step 9: Check if condition with the result of calling the method isPalindrome.Goto step 10.If true,goto step 16 else goto step 18

step 10: Check while loop condition.If true,goto step 11 else goto step

step 11: Check if condition.If true,goto step 12 else goto step 13

step 12: Return false value and goto step 16

step 13: Increment low value by 1 and goto step 14

step 14: Decrement high value by 1 and repeat step 10

step 15: Return true value and goto step 16

step 16: Return from ispalindrome method and goto step 17

step 17: Return low value and goto step 29

step 18: Check if condition with the result of calling the method isPalindrome.Goto step 19.If true,goto step 26 else goto step 27

step 19: Check while loop condition.If true,goto step 11 else goto step 28

step 20: Check if condition.If true,goto step 21 else goto step 22

step 21: Return false value and goto step 25

step 22: Increment low value by 1 and goto step 23

step 23: Decrement high value by 1 and repeat step 19

step 24: Return true value and goto step 25

step 25: Return from ispalindrome method and goto step 26

step 26: Return high value and goto step 29

step 27: Return -1 value and goto step 29

step 28: Return -2 and goto step 29

step 29: Check if statement condition.If true,goto step 30 else goto step 31

step 30: Print the statement and goto step 34

step 31: Check else if condition.If true,goto step 32 else goto step 33

step 32: Print statement and goto step 34

step 33: Print idx value and goto step 34

step 34: STOP

Program

**public class** RemovePalindrome {  
**static boolean** isPalindrome(String str,  
**int** low, **int** high)  
 {  
**while** (low < high)  
 {  
**if** (str.charAt(low) != str.charAt(high))  
**return false**;  
 low++;  
 high--;  
 }  
**return true**;  
 }  
  
*// This method returns -1 if it is  
 // not possible to make string a palindrome.  
 // It returns -2 if string is already  
 // a palindrome. Otherwise it returns  
 // index of character whose removal can  
 // make the whole string palindrome.***static int** possiblePalinByRemovingOneChar(String str)  
 {  
  
*// Initialize low and right  
 // by both the ends of the string***int** low = 0, high = str.length() - 1;  
  
*// loop untill low and  
 // high cross each other***while** (low < high)  
 {  
  
*// If both characters are equal then  
 // move both pointer towards end***if** (str.charAt(low) == str.charAt(high))  
 {  
 low++;  
 high--;  
 }  
**else**{  
  
*/\*  
 \* If removing str[low] makes the  
 \* whole string palindrome. We basically  
 \* check if substring str[low+1..high]  
 \* is palindrome or not.  
 \*/***if** (*isPalindrome*(str, low + 1, high))  
**return** low;  
  
*/\*  
 \* If removing str[high] makes the whole string  
 \* palindrome. We basically check if substring  
 \* str[low+1..high] is palindrome or not.  
 \*/***if** (*isPalindrome*(str, low, high - 1))  
**return** high;  
**return** -1;  
 }  
 }  
  
*// We reach here when complete string  
 // will be palindrome if complete string  
 // is palindrome then return mid character***return** -2;  
 }  
  
*// Driver Code***public static void** main(String[] args)  
 {  
 String str = **"abeceba"**;  
**int** idx = *possiblePalinByRemovingOneChar*(str);  
  
**if** (idx == -1)  
 System.***out***.println(**"Not Possible"**);  
**else if** (idx == -2)  
 System.***out***.println(**"Possible without "** +  
**"removing any character"**);  
**else**System.***out***.println(**"Possible by removing"** +  
**" character at index "** + idx);  
 }  
}

O/P

Possible without removing any character

**30.Program to remove duplicates from a string**

Algorithm

step 1:START

step 2: Read string and goto step 3

step 3: Assign string length to n and goto step 4

step 4: Call removeDuplicate method and print the result obtained from going to step 5.Goto step 13

step 5: Assign index to 0 and goto step 6

step 6: Check for loop condition.If true,goto strp 7 else goto step 12

step 7: Check for loop condition.If true,goto step 8 else goto step 10

step 8: Check if statement condition.If true,goto step 9 else repeat step 7

step 9: Break the loop and goto step

step 10: Check if statement condition.If true,goto step11 else repeat step 6

step 11: Assign str[i] to str[index++] and repeat step 6

step 12: Return the value of String.valueOf(Arrays.copyOf(str, index)) and goto step 4.

step 13: STOP

Program

**import** java.util.Arrays;  
  
**public class** RemoveDuplicates {  
**static** String removeDuplicate(**char** str[], **int** n)  
 {  
*// Used as index in the modified string***int** index = 0;  
  
*// Traverse through all characters***for** (**int** i = 0; i < n; i++)  
 {  
  
*// Check if str[i] is present before it***int** j;  
**for** (j = 0; j < i; j++)  
 {  
**if** (str[i] == str[j])  
 {  
**break**;  
 }  
 }  
  
*// If not present, then add it to  
 // result.***if** (j == i)  
 {  
 str[index++] = str[i];  
 }  
 }  
**return** String.*valueOf*(Arrays.*copyOf*(str, index));  
 }  
  
*// Driver code***public static void** main(String[] args)  
 {  
**char** str[] = **"reeepeaaat"**.toCharArray();  
**int** n = str.**length**;  
 System.***out***.println(*removeDuplicate*(str, n));  
 }  
}

O/P

Repat

**31.Program to return maximum occurring charcter to an input string**

Algorithm:

step 1:START

step 2: Read string and goto step 3

step 3: Print the result obtained by calling the method getMaxOccuringChar and following the step 4.Goto step 14

step 4: Assign value to character array and goto step 5

step 5: Check for loop condition.If true,goto step 6 els egoto step 7

step 6: Increment the value of count[str.charAt(i)] and repeat step 5

step 7: Initiaalixe max value to -1 and goto step 8

step 8: Initialise result value to space and goto step 9

step 9: Check for loop conditon.If true,goto step 10 else goto step 13

step 10: Check if statement condition.If true,goto step 11 else repeat step 9

step 11: Assign count[str.charAt(i)] value to max and goto step 12

step 12: Assign str.charAt(i) to result and repeat step 9

step 13: Return result value to step 3.

step 14: STOP

Program

**public class** MaxOccurChar {  
**static final int *ASCII\_SIZE*** = 256;  
**static char** getMaxOccuringChar(String str)  
 {  
*// Create array to keep the count of individual  
 // characters and initialize the array as 0***int** count[] = **new int**[***ASCII\_SIZE***];  
  
*// Construct character count array from the input  
 // string.***int** len = str.length();  
**for** (**int** i=0; i<len; i++)  
 count[str.charAt(i)]++;  
  
**int** max = -1; *// Initialize max count***char** result = **' '**; *// Initialize result  
  
 // Traversing through the string and maintaining  
 // the count of each character***for** (**int** i = 0; i < len; i++) {  
**if** (max < count[str.charAt(i)]) {  
 max = count[str.charAt(i)];  
 result = str.charAt(i);  
 }  
 }  
  
**return** result;  
 }  
  
*// Driver Method***public static void** main(String[] args)  
 {  
 String str = **"sample string"**;  
 System.***out***.println(**"Max occurring character is "** +  
*getMaxOccuringChar*(str));  
 }  
}

O/P

Max occurring character is s

**32.Program to print duplicates in a string**

Algorithm:

step 1:START

step 2: Assign string value and goto step 3

step 3: Call printDups method and goto step 4

step 4: Call fillCharCounts method and goto step 5

step 5: Check for loop condition.If true,goto step6 else goto step 7

step 6: Increment the value of count[str.charAt(i)] value to 1 and repeat step 5

step 7: Return from fillCharCounts method and goto step 8

step 8: Check for loop condition.If true,goto step 9 else goto step 11

step 9: Check if statement condition.If true,goto step 10 else repeat step 8

step 10: Print i and count[i] values and repeat step 8

step 11: Return from printDups method and goto step 12

step 12: STOP

Program

**public class** PrintDuplicates {  
**static final int *NO\_OF\_CHARS*** = 256;  
  
*/\* Fills count array with frequency of characters \*/***static void** fillCharCounts(String str, **int**[] count)  
 {  
**for** (**int** i = 0; i < str.length(); i++)  
 count[str.charAt(i)]++;  
 }  
  
*/\* Print duplicates present in the passed string \*/***static void** printDups(String str)  
 {  
*// Create an array of size 256 and fill count of every character in it***int** count[] = **new int**[***NO\_OF\_CHARS***];  
*fillCharCounts*(str, count);  
  
**for** (**int** i = 0; i <***NO\_OF\_CHARS***; i++)  
**if**(count[i] >1)  
 System.***out***.printf(**"%c, count = %d \n"**, i, count[i]);  
  
 }  
  
*// Driver Method***public static void** main(String[] args)  
 {  
 String str = **"test string"**;  
*printDups*(str);  
 }  
}

O/P

s, count = 2

t, count = 3

**33.Program to reverse words in a given string**

Algorithm

step 1:START

step 2: Read string value and goto step 3

step 3: Check for loop condition.If true,goto step 4 else goto step 5

step 4: Concat the strings ans,s[i] and space and repeat step 3

step 5: Print in next line and goto step 6

step 6: Print ans.substring(0, ans.length() - 1) value in next line and goto step 7

step 7: STOP

Program

**public class** ReverseString {  
**public static void** main(String[] args)  
 {  
 String s[] = **"i like this program very much"**.split(**" "**);  
 String ans = **""**;  
**for** (**int** i = s.**length** - 1; i >= 0; i--) {  
 ans += s[i] + **" "**;  
 }  
 System.***out***.println(**"Reversed String:"**);  
 System.***out***.println(ans.substring(0, ans.length() - 1));  
 }  
}

O/P

Reversed String:

much very program this like i

**34.Program to print longest prefix of a string which is also a suffix of the string**

Program

**public class** LongestPrefix {  
**static int** LengthlongestPrefixSuffix(String s)  
 {  
**int** n = s.length();  
  
**int** lps[] = **new int**[n];  
  
*// lps[0] is always 0*lps[0] = 0;  
  
*// Length of the previous  
 // longest prefix suffix***int** len = 0;  
  
*// Loop to calculate lps[i]  
 // for i = 1 to n - 1***int** i = 1;  
**while** (i < n)  
 {  
**if** (s.charAt(i) == s.charAt(len))  
 {  
 len++;  
 lps[i] = len;  
 i++;  
 }  
**else**{  
  
*// This is tricky. Consider  
 // the example. AAACAAAA  
 // and i = 7. The idea is  
 // similar to search step.***if** (len != 0)  
 {  
 len = lps[len - 1];  
  
*// Also, note that we do  
 // not increment i here*}  
  
*// If len = 0***else**{  
 lps[i] = 0;  
 i++;  
 }  
 }  
 }  
  
**int** res = lps[n - 1];  
  
*// Since we are looking for  
 // non overlapping parts***return** (res > n / 2) ? n / 2 : res;  
 }  
  
*// Function that returns the prefix***static** String longestPrefixSuffix(String s)  
 {  
*// Get the length of the longest prefix***int** len = *LengthlongestPrefixSuffix*(s);  
  
*// Stores the prefix*String prefix = **""**;  
  
*// Traverse and add charcaters***for** (**int** i = 0; i < len; i++)  
 prefix += s.charAt(i);  
  
*// Returns the prefix***return** prefix;  
 }  
  
*// Driver code***public static void** main(String[] args)  
 {  
 String s = **"abcab"**;  
 String ans = *longestPrefixSuffix*(s);  
**if** (ans == **""**)  
 System.***out***.println(**"-1"**);  
**else**System.***out***.println(ans);  
 }  
}

O/P

ab

**35.Program to print lexicographic rank of the string**

Algorithm:

step 1:START

step 2: Read string value and goto step 3

step 3: Call findRank method and goto step 4

step 4: Assign string length to len and goto step 5

step 5: Call fact method and goto step 6

step 6: Check the condition.If true,goto step 7 else repeat step 5

step 7: Return 1 and goto step 8

step 8: Return from fact method and assign the value to mul.Goto step 9

step 9:Assign 1 to rank and goto step 10

step 10: Check for loop condition.If true,goto step 11 else goto step

step 11: Divide mul with len-1 and assign to mul.Goto step 12

step 12: Call findSmallerInRight method and goto step 13

step 13: Assign countRight to 0 and goto step 14

step 14: Check for loop condition.If true,goto step 15 else goto step 17

step 15: Check if statement condition.If true,goto step 16 else repeat step 14

step 16: Increment countRight value by 1 and repeat step 14

step 17: Return countRight value and goto step 18

step 18: Return from findSmallerInRight method and assign result to countRight.Goto step 19

step 19: Add rank and countRight\*mul value and assign it to rank variable.Repeat step 10

step 20: Return rank value and goto step 21

step 21: Return from method firstRank and print the result.Goto step 22

step 22: STOP

Program

**public class** LexicographicRank {  
**static int** fact(**int** n)  
 {  
**return** (n <= 1) ? 1 : n \* *fact*(n - 1);  
 }  
  
*// A utility function to count smaller  
 // characters on right of arr[low]***static int** findSmallerInRight(String str, **int** low,  
**int** high)  
 {  
**int** countRight = 0, i;  
  
**for** (i = low + 1; i <= high; ++i)  
**if** (str.charAt(i) < str.charAt(low))  
 ++countRight;  
  
**return** countRight;  
 }  
  
*// A function to find rank of a string in  
 // all permutations of characters***static int** findRank(String str)  
 {  
**int** len = str.length();  
**int** mul = *fact*(len);  
**int** rank = 1;  
**int** countRight;  
  
**for** (**int** i = 0; i < len; ++i) {  
 mul /= len - i;  
  
*// count number of chars smaller  
 // than str[i] from str[i+1] to  
 // str[len-1]*countRight = *findSmallerInRight*(str, i, len - 1);  
  
 rank += countRight \* mul;  
 }  
  
**return** rank;  
 }  
  
*// Driver program to test above function***public static void** main(String[] args)  
 {  
 String str = **"string"**;  
 System.***out***.println(*findRank*(str));  
 }  
}

O/P

598