1. Which of these operators can be used to concatenate two or more String objects?

a) +

b) +=

c) &

d) ||

Answer: a

2. Which of these method of class String is used to obtain length of String object?

a) get()

b) Sizeof()

c) lengthof()

d) length()

Answer: d

3. Which of these method of class String is used to extract a single character from a String object?

a) CHARAT()

b) chatat()

c) charAt()

d) ChatAt()

Answer: c

4. Which of these constructors is used to create an empty String object?

a) String()

b) String(void)

c) String(0)

d) None of the mentioned

Answer: a

5. What is the output of this program?

```
class String_demo
{
    public static void main(String args[])
    {
        char chars[] = {'a', 'b', 'c'};
        String s = new String(chars);
        System.out.println(s);
    }
}
```

a) a

b) b

c) c

d) abc

Answer: d

6. What is the output of this program?

```
class StringDemo
{
    public static void main(String args[])
    {
        int ascii[] = { 65, 66, 67, 68};
        String s = new String(ascii, 1, 3);
        System.out.println(s);
    }
}
```

a) ABC

b) BCD

c) CDA

d) ABCD

Answer: b

7. What is the output of this program?

```
class String_demo
{
    public static void main(String args[])
    {
        char chars[] = {'a', 'b', 'c'};
        String s = new String(chars);
        String s1 = "abcd";
        int len1 = s1.length();
        int len2 = s.length();
        System.out.println(len1 + " " + len2);
    }
}
```

a) 3 0

b) 0 3

c) 3 4

d) 4 3

Answer: d

8. The output of the following fraction of code is

```
public class Test{
        public static void main(String args[]){
                String s1 = new String("Hello");
                 String s2 = new String("Hellow");
                 System.out.println(s1 = s2);
         }
}
```

a) Hello

b) Hellow

c) Compilation error

d) Exception

Answer: b

9. What will be the output of the following program?

```
public class Test{
        public static void main(String args[]){
                String s1 = "java";
                String s2 = "java";
                System.out.println(s1.equals(s2));
                System.out.println(s1 == s2);
        }
}
```

a) false true

b) false true

c) true false

d) true true

Answer: d

10. Determine output:

```
public class Test{
	public static void main(String args[]){
		String s1 = "SITHA";
		String s2 = "RAMA";
		System.out.println(s1.charAt(0) > s2.charAt(0));
	}
}
```

a) true

b) false

c) 0

d) compilation error

Answer: a

11. What could be output of the following fragment of code?

```
public class Test{
	public static void main(String args[]){
		String x = "hellow";
		int y = 9;
		System.out.println(x += y);
	}
}
```

a) compilation error

b) hellow9

c) 9hellow

d) hellow

Answer: b

12. toString() method is defined in

a) java.lang.String
b) java.lang.Object
c) java.util.ArrayList
d) java.io.Scanner

Answer: b

13. The String method compareTo() returns

a) true

b) false

c) int

d) -1

Answer: c

14. What could be output of the following fragment of code?

```
class Demo {

        public static void main(String args[]) {

                String s1="java";
                s1.concat(" rules");
                System.out.println(s1);

                 }
 }
```

a) java

b) java rules

c) rules java

d) none of these

Answer: a

15. What could be output of the following fragment of code?

```
class Demo {
        public static void main(String args[]) {
                StringBuffer s1=new
StringBuffer("java");
                s1.append(" rules");
                System.out.println(s1);
                 }
 }
```

a) java

b) java rules

c) rules java

d) none of these

Answer: b

Find output

```
String s1="java";
s1.reverse();
System.out.println(s1);
```

Output

- Compilation error

Find Output

```
public void testIfA() {
 if (testIfB("True")) {
 System.out.println("True");
 } else {
 System.out.println("Not true");
 }
 }
 public Boolean testIfB(String str) {
 return Boolean.valueOf(str);
 }
```

Output

- True

Find Output

String name="raju";

int day=25,month=4,year=1995;

System.***out.println(day+month+year+name);***

System.***out.println("welcome name "+day+month+year);***

System.***out.println("welcome "+name+day+month+year);***

Output

- 2024raju
- welcome name 2541995
- welcome raju2541995

Find Output

String name="welcome to java program";

System.***out.println(name.substring(8, 18)+name.charAt(11)+name.endsWith("ram"));***

Output

to java prjtrue

Find output:

```
String s1 = "raju";
StringBuffer s2 = new StringBuffer("raju");
System.out.println(s1.equals(s2));
```

Output:

false

Find Output:

```
System.err.print("hello");
```

Output:

hello

Find Output:

```
String str1 = "java";
char arr[] = { 'j', 'a', 'v', 'a', ' ', 'p',
'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g' };
String str2 = new String(arr);
System.out.println(str1);
System.out.println(str2);
```

output

java

java programming

Find Output:

```
String str1 = "";
char arr[] = { 'j', 'a', 'v', 'a', ' ', 'p',
'r', 'o', 'g', 'r', 'a', 'm', 'm', 'i', 'n', 'g' };
String str2 = new String(arr);
String str3 = new String(str2);

System.out.println(str1);
System.out.println(str2);
System.out.println(str3);
```

output

java programming

java programming

Find Output:

```
byte[] arr = { 97, 98, 99, 100, 101 };
String str2 = new String(arr);

System.out.println(str2);
```

output

abcde

Find Output:

```
String str = "Java Programming";
char arr[] = new char[10];
str.getChars(0, 4, arr, 0);
System.out.println(arr);
```

Output:

Java

Explanation:

The syntax of the method is: getChars(startindex, numOfCharacters, arrayName, startindexOfArrat). So from the string, starting from 0th index, first four characters are taken.

Find output

```
String str = "Java Programming";
char arr[] = new char[20];
arr = str.toCharArray();
System.out.println(arr);
```

output

Java Programming

- **Explanation:**
  toCharArray() method converts the string into a character array.

Find output

```
String str = "Java Programming";
String str1 = "Java Programming";

String str2 = str1;
if (str.equals(str1))
        System.out.println("Equal Case 1");
if (str == str1)
        System.out.println("Equal Case 2");
```

output

Equal Case 1

Equal Case 2

Explanation:
The equals() method compares the characters inside a String object.Thus str.equals(str1) comes out to be true.
The == operator compares two object references to see whether they refer to the same instance. Now str points to “Java Programming” and then str1 also points to “Java Programming”, hence str == str1 is also true.

Find output

```
String str = "Java Programming";
String str1 = "Programmingggggg";

if (str.regionMatches(5, str1, 0, 11))
        System.out.println("Same");
```

output

Same

Explanation:

The syntax of the function is: regionMatches(startIndex, stringS, stringS's_startIndex, numOfCharacters)

so starting from index 5, string str1 is compared from index 0 till 11 characters only.

Hence output is 'Same'

Find output

```
String str = "Java Programming";
String str1 = "Java";
if (str.startsWith("J"))
        System.out.println("Start Same");

if (str.endsWith("T"))
        System.out.println("End Same");
```

output

Start Same

Explanation:

The startsWith() method determines whether a given String begins with a specified string.
The endsWith() determines whether the String in question ends with a specified string.

Find output

```
String str = "JavaProgramming";
String str1 = "Java";

System.out.println(str.compareTo(str1));
System.out.println(str1.compareTo(str));
```

output

11

-11

Explanation:
The String method compareTo( ) serves the purpose of comparing two strings. Whether one string is less than, greater than or equal to the second string.
In case 1, comparing JavaProgramming with Java implies JavaProgramming is greater than Java by 11 characters.
In case 2, comparing Java with JavaProgramming implies Java is lesser than JavaProgramming by 11 characters (-11).

Find output

```
String str = "Java Programming";
String str1 = "Java";

System.out.println(str.indexOf("a"));
System.out.println(str.indexOf("m"));

System.out.println(str.lastIndexOf("a"));
System.out.println(str.lastIndexOf("m"));
```

output

1

11

10

12

Explanation:
indexOf( ) Searches for the first occurrence of a character or substring. lastIndexOf( ) Searches for the last occurrence of a character or substring.

Find output

```
String str = "Java Programming";
String str1 = "Java";
String str2 = str.substring(5);
System.out.println(str2);
System.out.println(str.substring(5, 9));
```

output

Programming

Prog

Explanation:
constructor substring(int startIndex) takes the substring starting from the specified index till end of the string
constructor substring(int startIndex, int endIndex) takes the substring from startIndex to endIndex-1

Find output

```
String str = "Java Programming";
System.out.println(str.replace('a', '9'));
```

output

J9v9 Progr9mming

Explanation:

The replace( ) method replaces all occurrences of one character in the invoking string with another character.

Hence ‘a’ replaced with a ‘9’.

Find output

```
String str = "Java";
String str1 = "Programming";
System.out.println(str.concat(str1));
```

output

JavaProgramming

Explanation:

concat() method simply concatenates one string to the end of the other

Find output

```
String GfG1 = "Welcome to java";
boolean GfG2;
GfG2 = GfG1.startsWith("hello");
System.out.println(GfG2);
```

output

a) true
b) false
c) 0
d) 1

Ans: b

Explanation: The startsWith() method is case sensitive “hello” and “Hello” are treated differently, hence false is stored in GfG2.

Find output

```
String GfG1 = "I am intern at xyz";
String GfG2 = new String(GfG1);
System.out.println((GfG1 == GfG2) + " " +
GfG1.equals(GfG2));
```

output

a) true true
b) false false
c) true false
d) false true

Output: d

Explanation: The == operator compares two object references to see whether they refer to the same instance, where as equals() compares the content of the two objects.

Find output

```
String GfG1 = "I am intern at GeeksforGeeks";
String GfG2 = new String(GfG1);
System.out.println((GfG1 == "I am intern at GeeksforGeeks") + " " + GfG1.equals(GfG2));
```

output

a) true true
b) false false
c) true false
d) false true

Output: a

Explanation: The == operator compares two object references to see whether they refer to the same instance but when using == with a string literal(not an instantiated String variable) will only compare the content of the strings.

Find output

```
String GfG[] = { "a", "b", "c", "a", "c" };
for (int i = 0; i < GfG.length; ++i)
        for (int j = i + 1; j < GfG.length; ++j)
                if (GfG[i].compareTo(GfG[j]) == 0)
                        System.out.print(GfG[j]);
```

output

a) ab
b) bc
c) ca
d) ac

Ans: d

Explanation: compareTo() function returns zero when both the strings are equal, it returns a value less than zero if the invoking string is less than the other string being compared and value greater than zero when invoking string is greater than the string compared to.

Find output

```
String GfG1 = "Hello";
String GfG2 = new String(GfG1);
String GfG3 = "HELLO";
System.out.println(GfG1.equals(GfG2) + " " +
GfG2.equals(GfG3));
```

output

a) true true
b) false false
c) true false
d) false true

Output: c

Explanation: As we know that equal() method compares the content of the strings. GfG1 and GfG are having the same content. But as we know equal() is case sensitive, therefore GfG2 and GfG3 are different.

Find output

```
StringBuffer GfG1 = new StringBuffer("Hello");
StringBuffer GfG2 = new StringBuffer(" World");
GfG1.append(GfG2);
System.out.println(GfG1);
```

output

a) Hello
b) World
c) Helloworld
d) Hello World

Output: d

Explanation: append() method of class StringBuffer is used to concatenate the string representation to the end of invoking string.

Find output

output

Find output

output