```
Predict the following output for the given code
public class TestApp{
      public static void main(String... args){
                  String[] arr[] ={{"%","***"},{"!!!!","@@@@@","######"}};

                  for(String str[] : arr){
                        for(String s:str)
                        {
                              System.out.println(s);
                              if(s.length()==4)
                                    break;
                        }
                        break;
                  }
      }
}
A. Compile Time Error
B. StringIndexOutOfBoundsException
C. %
   ***
D. %
   ***
   ######
E. ArrayIndexOutOfBoundsException
E. None of the above

Answer : C

Q>
What will be the result of compiling and executing Test class?
public class Test {
    public static void main(String[] args) {
        int score = 60;
        switch (score) {
            default:
                System.out.println("Not a valid score");
            case score < 70:
                System.out.println("Failed");
                break;
            case score >= 70:
                System.out.println("Passed");
                break;
        }
    }
}
A. Compilation Error
B. Failed
C. Not a valid score
   Failed
D. Passed

Answer : A

Q>
What will be the result of compiling and executing Test class?
public class Test {
    public static void main(String[] args) {
        int i = 5;
        if(i++ < 6) {
            System.out.println(i++);
        }
    }
}
A. 5
B. 6
C. 7
D. Nothing printed on the console

Q>
Consider below code:
//Test.java
public class Test {
    private static boolean flag = !true; // false

    public static void main(String [] args) {
        System.out.println(!flag ? args[0] : args[1]);//true ? AM : PM ---> AM
    }
}
What will be the result of compiling and executing Test class using below commands?
javac Test.java
java Test AM PM
      |
    Test.main(new String[]{"AM","PM"}

A. AM
B. PM
C. ExceptionInitalizerError while loading the .class file
D. CompilationError

Q>
What will be the result of compiling and executing Test class?
public class Test {
    public static void main(String [] args) {
        int a = 100;
        System.out.println(-a++);
    }
}
A. CompilationError
B. -100
C. -101
D. 99
E. -99

Answer: B

Q>
What will be the result of compiling and executing Test class?
public class Test {
    public static void main(String [] args) {
        int a = 2;
        boolean res = false;
        res = a++ == 2 || --a == 2 && --a == 2;
        System.out.println(a);
    }
}
A. 2
B. 3
C. 1
```

D. CompilationError

Answer: B

Method Overloading
+++++++++++++++++++
 => Two or more methods is said to be overloaded method, iff both the methods have same name,but change in the argument datatype.
 => Order of binding the method calls would be based on
      a. exact match
      b. type promotion
=> In case of reference type, first priority is for child and then for parent.
=> In case of Overloading,binding the method call will be done by the compiler based on the reference type.
=> It is also called as "CompileTime Polymorphism/Early Binding".

```
eg#1.
class Test
{
      public void m1(int i,float f)
      {
            System.out.println("int-float arg method");
      }
      public void m1(float f,int i)
      {
            System.out.println("float-int arg method");
      }
      public static void main(String[] args)
      {
            Test t= new Test();
            t.m1(10,10.5f);//int-float
            t.m1(10.5f,10);//float-int
            t.m1(10,10);  //CE: ambigous
            t.m1(10.5f,10.5f);//CE: can't find symbol
      }
}
```

Var-Args(Variable no of arguments method)
+++
 -> Untill 1.4V we can't declare a method with variable no of arguments
 -> if there is a change in no of arugments compulsorily we have to define a new method
 -> This approach increases the length of the code and reduces the readability.
 -> From 1.5V version onwards we can declare a method with variable no of arugments such type of methods are called as "VAR-ARGS METHODS".
 -> Syntax:

```
      public XXXXX methodName(XXXX... varaible)
      {

      }
```

```
eg#1.
class Test{
      public void m1(int... data)
      {
            System.out.println("Var-Arg method");
      }
      public static void main(String[] args) {
```

```
                  Test t= new Test();
                  t.m1();
                  t.m1(10);
                  t.m1(10,20);
                  t.m1(10,20,30);
                  t.m1(10,20,30,40);
      }
}
```

Note: Internally var-arg parameter is implemented by using "1-D Array",so var-arg parameter can be accessed through index.

eg#2.

```
class Test{
      public void m1(int... data)
      {
            int total = 0;
            for (int i =0;i< data.length ;i++ )
            {
                  System.out.print("data["+ i +"]="+data[i] + "\t");
                  total = total + data[i];

            }
            System.out.print("Total = "+total);
            System.out.println();
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1();
                  t.m1(10);
                  t.m1(10,20);
                  t.m1(10,20,30);
                  t.m1(10,20,30,40);
      }
}
```

Output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
Total = 0
data[0]=10      Total = 10
data[0]=10      data[1]=20      Total = 30
data[0]=10      data[1]=20      data[2]=30      Total = 60
data[0]=10      data[1]=20      data[2]=30      data[3]=40      Total = 100
```

eg#3.

```
class Test{
      public void m1(int... arr){ // int[] arr = new int[0]  , int[] arr = new 
int[1] , int[] arr = new int[2],....
            int total = 0;
            for(int data : arr){
                  total += data;
            }
            System.out.println(total);
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1();
                  t.m1(10);
                  t.m1(10,20);
                  t.m1(10,20,30);
                  t.m1(10,20,30,40);
      }
}
Output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
0
10
30
60
100
```

Case1:
Which of the following var-arg declarations are valid?

```
  methodOne(int... x) //valid [recomended]
  methodOne(int ...x) //valid
  methodOne(int...x)  //valid
  methodOne(int x...) //invalid
  methodOne(int. ..x) //invalid
  methodOne(int .x..) //invalid
```

Case2: we can mix var-args with normal parmeters also, and normal parameter can be different type and var-arg can be different type.

```
      eg:: m1(int a, int... arr)
           m1(String name, int... arr)
class Test{
      public void m1(String name, int... arr){
            System.out.println(name);
            System.out.println(arr);
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1("sachin",20,30,40);

      }
}
Output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
sachin
[I@76ed5528
```

Case3:We can mix var-arg parameter with normal parameter,but in the parameter list the var-arg parameter should be at the last.

```
class Test{
      public void m1(int... arr, int data ){
            System.out.println(data);
            System.out.println(arr);
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
      }
}
Output
D:\OctBatchMicroservices>javac Test.java
Test.java:2: error: varargs parameter must be the last parameter
```

```
        public void m1(int... arr, int data ){
```

Case4: In a paramter list,we can have only var-arg parameters,more than one results in "CompiletimeError".

```
class Test{
      public void m1(int... arr1, int... arr2 ){
            System.out.println(arr1);
            System.out.println(arr2);
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
      }
}
```

Output

```
D:\OctBatchMicroservices>javac Test.java
Test.java:2: error: varargs parameter must be the last parameter
        public void m1(int... arr1, int... arr2 ){
```

Case5:In general var-arg method will get least priority that is if no other methods are available to bind only then var-arg method will get
      a chance for binding. This is just like "default" statement in switch case.

```
class Test{
      public void m1(int... arr ){// arr-> 0,1,.... n
            System.out.println("var-arg method");
      }
      public void m1(int i){// i -> 1
            System.out.println("one-arg method");
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
                  t.m1(10);
                  t.m1();
      }
}
```

output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
var-arg method
one-arg method
var-arg method
```

Case6: For the var-args method we can provide the corresponding type array as argument

```
class Test{
      public void m1(int... arr ){//int[] ::  arr-> 0,1,.... n
            System.out.println("var-arg method");
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
                  t.m1(new int[]{100,200,300,400});
      }
}
```

output

```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
var-arg method
```

var-arg method

Case7:
```
class Test{
      public void m1(int... arr ){//m1(int[]) ::  arr-> 0,1,.... n
            System.out.println("Var-arg method");
      }
      public void m1(int[] arr) //m1(int[])
      {
            System.out.println("Array-arg method");
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
                  t.m1(new int[]{100,200,300,400});
      }
}
output
D:\OctBatchMicroservices>javac Test.java
Test.java:5: error: cannot declare both m1(int[]) and m1(int...) in Test
        public void m1(int[] arr)//m1(int[])
                    ^
1 error
```

Case 8: wherever there is [] array,we can replace it with "..." also to provide the arguments in flexible manner(var-args,arrays)
Case 9: Wherever theer is ..., if we repalce it "[]", we don't get flexibility to provide the arguments in var-args and arrays style.

eg::
```
class Test{
      public void m1(int... arr ){// arr-> 0,1,.... n
            System.out.println("Var-arg method");
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
                  t.m1(new int[]{100,200,300,400});
      }
}
output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
Array-arg method
Array-arg method
```

eg::
```
class Test{
      public void m1(int[] arr ){// arr-> 0,1,.... n
            System.out.println("Array-arg method");
      }
      public static void main(String[] args) {
                  Test t= new Test();
                  t.m1(10,20,30,40);
                  t.m1(new int[]{100,200,300,400});
      }
}
D:\OctBatchMicroservices>javac Test.java
Test.java:7: error: method m1 in class Test cannot be applied to given types;
```

```
                        t.m1(10,20,30,40);


eg::
class Test{
      public static void main(String... args) {
            System.out.println("Var-Arg main method");
      }
}
output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
Var-Arg main method

Case 8:
class Test{
      public void m1(int[]... twoDarr){
            System.out.println(twoDarr);
            for (int[] oneDArr: twoDarr )
            {
                  for (int data: oneDArr )
                  {
                        System.out.print(data+"\t");
                  }
                  System.out.println();
            }
      }
      public static void main(String... args) {
            Test t =new Test();
             int[] arr1 = {10,20,30};
             int[] arr2 = new int[]{100,200,300};
             t.m1(arr1,arr2);
      }
}
//m1(int... x)   ===> arguments :: var-args,array ---> x :: []
//m1(int[]... x) ===> arguments :: 1D-array var-args  ----> x ::[][]

Output
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
[[I@76ed5528
10      20      30
100     200     300
```

Rule1: Binding of method call will happen based on the reference, not on the runtime object

```
class Animal{}
class Monkey extends Animal{}
class Test{
      public void talk(Monkey m){
            System.out.println("Monkey version");
      }
      public void talk(Animal a){
            System.out.println("Animal version");
      }
      public static void main(String... args) {
            Test t = new Test();
            Animal a =new Animal();
            t.talk(a);//Animal version

            Monkey m =new Monkey();
            t.talk(m);//Monkey version

            Animal a1= new Monkey();// a1 -> Animal type(compiler will bind)  a1->
Monkey(runtime object::JVM)
            t.talk(a1);//Animal version

      }
}
```
Output
```
D:\OctBatchMicroservices>javac Test.java
D:\OctBatchMicroservices>java Test
Animal version
Monkey version
Animal version
```

Try it yourself
+++++++++++++++
What will be the result of compiling and executing Test class?
```
public class Test {

    public static void main(String [] args) {
        int a = 3;// a = 3,4,5
        m(++a, a++); // m(4,4) //pass by value
        System.out.println(a);//5
    }

    private static void m(int i, int j) {
        i++;
        j--;
    }
}
```
A. 4
B. 5
C. 6
D. 3

Answer: B

++++++++++++++++++++
Important Discussion
++++++++++++++++++++
Strings in Java
  => String in java refers to Collection of Characters.

When we can Store collection of characters in char[], why do we need String class in java?
 => if we use String as Array of charactes, then as a java programmer to perform some operation on the array data we need to write a method
 => Writing up a logic and working with that logic is always tough for programmer.
 => To reduce the burden for a developer, SUNMS team had made "Array of characters" data as "Object".
 => Since they made Array of characters as "Object", we need a blue print for an Object called "class" and this blueprint is only "String".

```
eg#1.
char[] name = {'s','a','c','h','i','n'};
System.out.println(name);//sachin
convertToUpper(name);
convertToLower(name);

public void convertToUpper(char[] name)
{
      //logic
}

public void convertToLower(char[]name)
{
      //logic
}
```

java.lang.String
================
String it refers to an Object in java present in package called java.lang.String
String refers to collection of characters

```
eg:: String s= "sachin";
     System.out.println(s);//sachin

     String s =new String("sachin");
     System.out.println(s);//sachin
```

In java String object is by default immutable,meaning once the object is created we cannot change the value of the object, if we try to change then those changes will be reflected on the new object not on the existing object.

```
case 1:: String s= "sachin";
         s.concat("tendulkar");(new object got created with modification so
immutable)
         System.out.println(s);

         output::sachin

                  vs

         StringBuilder sb=new StringBuilder("sachin");
         sb.append("tendulkar");(on the same object modification so mutable)
         System.out.println(sb);

         output:: sachintendulkar
```

Note:

```
public class Object{
      public boolean equals(String data){
            Compares the reference
                  if both are equal  -> return true
                  otherwise -> return false
      }
      public String toString(){
            print the address of the object
      }
}

public final class String extends Object{
      public String concat(String data){}
      public String toUpperCase(String data){}
      public String toLowerCase(String data){}

      @Override
      public boolean equals(String data){
            compares the content present inside the object
            if both the data are equal -> return true
            otherwise -> return false
      }

      @Override
      public String toString(){
            //prints the data present inside the object
    }
}
public final class StringBuffer extends Object{

      public String append(String data){}
      public String toUpperCase(String data){}
      public String toLowerCase(String data){}

      //use it from object class
      public boolean equals(String data){
            Compares the reference
                  if both are equal  -> return true
                  otherwise -> return false
      }

      @Override
      public String toString(){
            //prints the data present inside the object
    }
}

case 2:: String s1 = new String("sachin");
                String s2 = new String("sachin");
         System.out.println(s1==s2); //false
         System.out.println(s1.equals(s2));//true
           => String class .equals method will compare the content of the object
                 if same return true otherwise return false

                      vs

         StringBuilder sb1 = new StringBuilder("sachin");
         StringBuilder sb2 = new StringBuilder("sachin");
         System.out.println(sb1==sb2); //false
         System.out.println(sb1.equals(s2));//false
            => StringBuilder class .equals method is not overriden so it will use
Object
               class .equals() which is meant for reference comparison.
               if differnt object returns false,even if the contents are same.


Snippets
++++++++
Consider below code of Test.java file:
public class Test {
    public static void main(String [] args) {
        boolean flag = false;
        System.out.println((flag = true) | (flag = false) || (flag = true));
        System.out.println(flag);
    }
}
What is the result of compiling and executing Test class?
A. true
   false
B. false
   true
C. true
   true
D. false
   false
E. CompilationError

Answer: A

Consider below code of Test.java file:
public class Test {
    public static void main(String [] args) {
        boolean status = true;
        System.out.println(status = false || status = true | status = false);
        System.out.println(status);
    }
}
What is the result of compiling and executing Test class?
A. true
   false
B. false
   true
C. true
   true
D. false
   false
E. CompilationError

Answer: E

Q>
Consider below code of Test.java file:
public class Test {
    public static void main(String [] args) {
        int a = 3;//2
        int b = 5;//4
        int c = 7;//7
        int d = 9;//9
        boolean res = --a + --b < 1 && c++ + d++ > 1;
        System.out.printf("a= "+a+",b= "+b+",c= "+c+",d="+d+",res="+res);
    }
}
A. a=2,b=4,c=7,d=9,res=false
B. a=2,b=4,c=8,d=10,res=false
C. a=2,b=4,c=7,d=9,res=true
D. a=2,b=4,c=8,d=10,res=true
E. a=3,b=5,c=8,d=10,res=false
F. a=3,b=5,c=8,d=10,res=true
```

Answer: A

```
res = --a + --b < 1  &&  c++ + d++ > 1;
    = 2   + 4 < 1    &&  c++ + d++ > 1
    =  6 < 1         &&  c++ + d++ > 1
    =  false &&
    =  false
```