＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
＊关于 Java 测试试题
＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊＊
問 1　运行下面的程序，选出一个正确的运行结果。


```
public class Sample {
      public static void main(String[] args) {
            int[] test = { 1, 2, 3, 4, 5 };
            for(int i = 1 ; i System.out.print(test[i]);
           }
      }
}
```

1. 编译错误。
2. 执行时发生异常抛出
3. 执行结果是「1234」。
4. 执行结果是「12345」
5. 执行结果是「234」
6. 执行结果是「2345」

問 2　运行下面的程序，选出一个正确的运行结果。

```
public class Sample {
      public static void main(String[] args) {
            int[] test1 = { 10, 20, 30 };

            int[] test2 = test1;

            test2[2] = test1[0];
```

```
        System.out.println(test1[0] + ":" + test1[1] + ":" + test1[2]);
    }
}
```


1. 编译时发生错误。
2. 执行时，抛出异常
3. 「10:20:30」显示出来
4. 「30:20:30」显示出来
5. 「10:20:10」显示出来
6. 上面以外的结果显示出来。

問 3　运行下面的程序，选出一个正确的运行结果。

```
public class Sample {
    public static void main(String[] args) {
        boolean[] b = new boolean[2];

        b[1] = !b[0];

        if (b[0]) {
            System.out.println("X");
        } else if (b[1]) {
            System.out.println("Y");
        } else {
            System.out.println("Z");
        }
    }
}
```

1. 编译错误。
2. 输出结果「X」。
3. 输出结果「Y」。
4. 输出结果「Z」。
5. 其他输出结果。

問 4 **运行**下面的程序，选出一个正确的运行结果。

```
public class Sample {
    public static void main(String[] args) {
        String[] arr = { "ABC", "DE", "FGHI" };

        System.out.print(arr.length);
        System.out.print(arr[1].length());
    }
}
```

1. 编译错误。
2. 运行时抛出异常。
3. 输出结果「22」。
4. 输出结果「23」。
5. 输出结果「32」。
6. 输出结果「33」。

問 5 **运行**下面的程序，选出一个正确的运行结果。

```
public class Sample {
    public static void main(String[] args) {
        int[] arr = { 0, 1, 1, 2, 2 };
```

```
        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[arr[i]]);
        }
    }
}
```


1. 编译错误。
2. 运行时抛出异常。
3. 输出结果「01122」。
4. 输出结果「00000」。
5. 输出结果「11111」。
6. 输出结果「22222」。
7. 输出结果「01111」。

問 6 选择出下面数组的初始化结果不正确的 3 个答案。

1. int[3] test;
2. int[][] test = new int[3][5];
3. int[][] test = new int[3][];
4. int[][] test = new int[][5];
5. int test[] = {1, 2, 3};
6. int[] test = new int[3]{1, 2, 3};
7. int[][] test = { {1, 2}, {3, 4, 5} };

問 7 **运行**下面的程序，选出一个正确的运行结果。

```
public class Sample {
    public static void main(String[] args) {
        int[][] arr = {
                { 1, 2 },
                { 3 },
                { 0, 1, 2, 3 }
```

```
            };

            int sum = 0;

            for (int i = 0; i < arr.length; i++) {
                sum += arr[i].length;
            }

            System.out.print(sum);
        }
}
```

1. 编译错误。
2. 运行抛出异常。
3. 输出结果「0」。
4. 输出结果「3」。
5. 输出结果「7」。
6. 输出结果「12」。

問　8　使用下面定义好的类和接口。从答案中选出可以正常编译的类。

```
class Super1{}
class Super2{}
interface IF1{}
interface IF2{}
```

1. class Sub extends Super1, Super2{}
2. class Sub implements IF2 extends Super1{}
3. class Sub extends IF1{}
4. class Sub extends Super1 implements IF1, IF2{}
5. interface IF3 extends IF1, IF2{}
6. interface IF3 implements IF1, IF2{}

問　9　运行下面的程序，选出一个正确的运行结果。

```
interface IF{
void show();
}
class X implements IF{
void show(){
System.out.print("X");
}
}
class Y extends X{
public void show(){
System.out.print("Y");
}
}
class Sample{
public static void main(String[] args){
IF obj = new Y();
obj.show();
}
}
```

1. ２行出现编译错误。
2. ４行出现编译错误
3. ５行出现编译错误
4. 16 行出现编译错误。
5. 16 行出现编译错误
6. 输出结果「X」。
7. 输出结果「Y」。

問　１０　运行下面的程序，选出一个正确的运行结果。（包含空白行）

```
interface Radius{
int R = 10;
}
interface Constant{
double PI = 3.14;
}
class Sample implements Constant{

public static void main(String[] args){

System.out.println(Radius.R * Radius.R * PI);

}
}
```

1. ２行出现编译错误。
2. 5 行出现编译错误。
3. 7 行出现编译错误。
4. 11 行出现编译错误。
5. 输出结果「314」。
6. 输出结果「314.0」。

問　１１　运行下面的程序，选出一个正确的运行结果。

```
interface IF{
void show();
}
abstract class Super implements IF{
protected void show(int i){
```

```
System.out.print("X");
}
}
class Sub extends Super{
public void show(){
show(10);
System.out.print("Y");
}
}
class Sample{
public static void main(String[] args){
IF obj = new Sub();
obj.show();
}
}
```

1. ４行出现编译错误。
2. ５行出现编译错误。
3. 11 行出现编译错误。
4. 17 行出现编译错误。
5. 18 行出现编译错误。
6. 输出结果「X」。
7. 输出结果「XY」。

問　１２　运行下面的程序，选出一个正确的运行结果。

```
interface IF{}
class Super{}
class Sub extends Super implements IF{}

public final class Sample{
```

```
public static void main(String[] args){

Sub w = new Sub();
Super x = (Super)w;
IF y = (IF)x;
Sample z = (Sample)y;
}
}
```

1. 10 行出现编译错误。
2. 11 行出现编译错误。
3. 12 行出现编译错误。
4. 13 行运行时抛出异常。
5. 11 行运行时抛出异常。
6. 12 行运行时抛出异常。
7. 正常编译，正常运行。

問　１３　运行下面的程序，选出一个正确的运行结果。（包含空白行）

```
interface IF{ }

class Super implements IF{ }

class Sub extends Super{

public IF test(){
return new Super();
}
}
public class Sample extends Sub{

public static void main(String[] args){
```


```
new Sub().test();
}
public Sub test(){
return (Sub)new Super();
}
}
```


1. ８行出现编译错误。
2. 14 行出现编译错误。
3. 16 行出现编译错误。
4. 17 行出现编译错误。
5. ８行运行时抛出异常。
6. 17 行运行时抛出异常。
7. 正常编译，正常运行。

問 １４　运行下面的程序，选出一个正确的运行结果。

```
abstract interface IF{
void test();
}
class X implements IF{
public void test(){
System.out.print("X");
}
}
class Y extends X implements IF{}
class Z extends Y{
public void test(){
System.out.print("Z");
}
}
```

```
class Sample{
public static void main(String[] args){
IF obj = new Z();
obj.test();
}
}
```

1. 1 行出现编译错误。
2. 9 行出现编译错误。
3. 17 行出现编译错误。
4. 18 行出现编译错误。
5. 输出结果「X」。
6. 输出结果「Z」。

問 １５　运行下面的程序，选出一个正确的运行结果。

```
interface IF1{
void test();
}
interface IF2{
void test();
}
class Test implements IF1{
public void test(){
System.out.print("OK");
}
}
class Sample{
public static void main(String[] args){
IF1 if1 = new Test();
if1.test();
IF2 if2 = (IF2)if1;
```

```
if2.test();
}
}
```

1. 15 行出现编译错误。
2. 16 行出现编译错误。
3. 17 行出现编译错误。
4. 15 行运行时抛出异常。
5. 16 行运行时抛出异常。
6. 17 行运行时抛出异常。

問　１６　以下的程序出现编译错误。请指出编译出错的行。

```
interface IFA{}
interface IFB extends IFA{}
interface IFC extends IFA{ void test(); }
interface IFD extends IFB, IFC{}
public class Sample implements IFD{
public void test(){}
public static void main(String... args){
IFD x = new Sample();
x.test();
IFC y = x;
y.test();
IFB z = (IFB)y;
z.test();
}
}
```

1. ８行
2. ９行

3. 10 行
4. 11 行
5. 12 行
6. 13 行

問 １７　请指出对于 JSP 的 web.xml 的使用方法正确的说明。

1. <servlet>的子要素<jsp-file>是用来指定 JSP 文件路径的。
2. <servlet>的子要素<servlet-name>是用来指定 JSP 文件路径的。
3. <servlet-mapping>的子要素<servlet-name>是用来指定 JSP 文件路径的。
4. <servlet-mapping>的子要素<url-pattern>是用来指定 JSP 文件路径的。

問 １８ javax.servlet.http.HttpSession 接口里选出可以实现从 session获取数据的方法。

1. getAttribute
2. setAttribute
3. getSession
4. setSession

問 １９ 请选出 javax.servlet.http.HttpServlet 类所拥有的方法。

1. doGet
2. destroy
3. getAttribute
4. doPost

問 ２０ Servlet 的初始参数通过 javax.servlet.http.HttpServlet 类的哪一种方法。

1. getDataByteArray
2. getServletConfig
3. getAttribute
4. getInitParameter