B

Test 1 `java D`

```
01  public class D {
02      static int x=3;
03      public static void main(String[] args) {
04         D a5=new D();      a5.f();
05         a5=new D();        a5.f();
06         System.gc();       System.runFinalization();
07      }
08      void f() {Pippo a=new Pippo2();
09      }
10      public void finalize() { System.out.print("X"); }
11      class Pippo {
12          int k;
13          Pippo() {k=++x;}
14          public void finalize() { System.out.print(k); }
15      }
16      class Pippo2 extends Pippo {
17              Pippo2() {k=x++;}
18      }}
```

Test 2 `java Sette`

```
01  public class Sette {
02      Sette(){
03          Collection<String> a = new ArrayList<String>();
04          Collection<String> b = new HashSet<String>();
05          for (int k=0;k<10;k++) {
06              String s="A"+(k%8);
07              a.add(s);
08              b.add(s);
09          }
10          System.out.println(a.size()+b.size());
11      }
12      public static void main(String[] a) {
13           new Sette();
14  }}
```

Test 3 `java Sette`

```
01  public class Sette {
02      Sette(){
03          Set<String> a = new ArrayList<String>();
04          Set<String> b = new HashSet<String>();
05          for (int k=0;k<10;k++) {
06              String s="A"+(k%8);
07              a.add(s);
08              b.add(s);
09          }
10          System.out.println(a.size()+b.size());
11      }
12      public static void main(String[] a) {
13           new Sette();
14  }}
```

B

Test 4

```
01  #include <cstdlib>
02  #include <iostream>
03  using namespace std;
04  void f(int k[0]){ --k[1]; }
05  void g(int * k){ (*(--k))++;}
06  void h(int k[1]){--k;}
07  int main() {
08      int k[]={1,2,3,4};
09      f(k+2);
10      g(k+2);
11      h(k+2);
12      for (int i=1;i<4;i++) { cout<<k[i]; }
13      return 0;
14  }
```

Test 5 java B

```
00  public class B {
01      static int x = 2;
02      B() { x++; }
03      B(int y) {x=x+y;}
04      public void finalize(){x--;}
05      public static void main(String[] args) {
06          A a2 = new A();
07          System.out.print(a2);
08          A a1 = new A(2);
09          System.out.print(a1);
10      }
11      public String toString(){return ""+x;}
12  }
13  class A extends B {
14      int x=4;
15      A() { x++; }
16      A(int x) {super(x); x++;}
17      public String toString(){return super.toString()+x;}
18  }
```

Test 6 java C

```
00  public class C {
01      int s=8;
02      void f() {System.out.print("A"+(++s));}
03      public static void main(String[]a){
04      C y=new C();
05      C x=new C() {
06          void f() {System.out.print("B"+(--s));}
07      };
08      x.f();
09      y.f();
10    }
11  }
```

B

Test 7 java uno.C

```
01 package uno;
02 public class C {
03   void f(int k) {
04     System.out.print(k*3);
05   }
06   public static void main (String args[]){
07     Object z = new B();
08     if (z instanceof uno.C) ((C) z).f(3);
09     if (z instanceof uno.B) ((B) z).f(6);
10   }
11   public static void main (String var){
12     Object z = new B();
13     if (z instanceof uno.C) ((C) z).f(3);
14   }}
15 class B extends C{
16   void f(int k) {
17     System.out.print(k*2);
18   }}
```

Test 8 java B1

```
01 public class A { // this class is in the A.java file
02     public int x=0;
03     private A() {x=3;}
04     A(int x) {x=6;}
05 }
06 public class B1 { // this class is in the B1.java file
07     public static void main(String a[]){
08         A a1=new A();
09         A a2=new A(9);
10         System.out.println(a1.x+" "+a2.x);
11     }
12 }
```

Test 9 – scrivere nel campo per l'output del test la sequenza risultante indicando V per le affermazioni vere e F per quelle false

9.1	Nei metodi statici non è possibile leggere e scrivere le variabili di istanza
9.2	Se una classe è astratta è permesso usarla per effettuare ereditarietà multipla
9.3	In una classe ci può essere un solo metodo public static void main
9.4	Se di un metodo f faccio overriding non è detto che ci siano più "signatures" (firme) di metodi legate al nome
9.5	Se di un metodo f faccio overloading non è detto che ci siano più "signatures" (firme) di metodi legate al nome
9.6	Il garbage collector di Java sospende l'esecuzione del programma finchè non ha finito di liberare la memoria.
9.7	In Java non esistono le variabili globali
9.8	Le variabili dichiarate static non sono modificabili dai metodi privati