Versione 1 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*x = y[4]$, sapendo che x è un puntatore e che y è un vettore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a))$ e $mem(env(y))$.

Versione 2 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*x = \&y$, sapendo che x è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $env(x)$ e $mem(mem(env(a)) + 1)$.

Versione 3 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*x) = y$, sapendo che x è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $env(x) + 4$ e $mem(mem(env(y)))$.

Versione 4 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $x[a] = *(a + z)$, sapendo che x è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(mem(env(a)) + mem(env(z)))$.

Versione 5 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(a + z) = *(*y)$, sapendo che y è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a)) + mem(env(z))$ e $mem(env(y))$.

Versione 6 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*a = \&(y[*p])$, sapendo che y è un vettore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a)) + mem(env(z))$ e $mem(env(y))$.

Versione 7 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $x = y$.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a)) + 2$ e $env(y) + 1$.

Versione 8 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*x) = y$, sapendo che x è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $env(x)$ e $mem(env(y) + mem(mem(env(p))))$.

Versione 9 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $x = \&y$.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(mem(env(y)))$.

Versione 10 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*p + 1) = *y$, sapendo che y è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $env(x) + mem(env(a))$ e $env(y) + mem(mem(env(p)))$.

Versione 1 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) Un attributo `static` non può essere acceduto mediante un riferimento a un oggetto della sua classe di appartenenza

B) Una classe non può essere dichiarata `private`

C) Un array non possiede dei membri

D) Un array con riferimento `null` ha lunghezza zero

E) Una classe può essere dichiarata `protected`

Versione 2 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) La dimensione di un array deve essere indicata al momento della dichiarazione dell'array

B) Una classe può essere dichiarata `private`

C) Quando due metodi hanno lo stesso nome si ha overloading

D) Un array con riferimento `null` ha lunghezza zero

E) I modificatori che precedono una variabile di tipo array si applicano alla variabile array ed anche ai suoi elementi

Versione 3 dell'esercizio 1

Dire quale delle seguenti affermazioni è falsa:

A) La dimensione di un array può non essere indicata al momento della dichiarazione dell'array

B) Un array con riferimento `null` non ha lunghezza

C) Un array vuoto può avere riferimento `null`

D) Tutti i tipi di eccezioni estendono la classe `Object`

E) È possibile dichiarare un array senza indicarne la dimensione

Versione 4 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) Tutti i tipi di eccezioni estendono la classe RuntimeException

B) Un array non possiede dei membri

C) Non è possibile dichiarare un attributo senza inizializzarlo

D) I modificatori che precedono una variabile di tipo array si applicano alla variabile array ed anche ai suoi elementi

E) Una classe può essere dichiarata protected

Versione 5 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) Una classe non può essere dichiarata `private`

B) Un array non possiede dei membri

C) Ogni valore in memoria è associato ad un tipo di dato particolare

D) Tutti i tipi di eccezioni estendono la classe `RuntimeException`

E) Non è possibile dichiarare un array senza indicarne la dimensione

Versione 6 dell'esercizio 1

Dire quale delle seguenti affermazioni è falsa:

A) La dimensione di un array può non essere indicata al momento della dichiarazione dell'array

B) Un array con riferimento `null` ha lunghezza zero

C) Un array vuoto non può avere riferimento `null`

D) Una classe può essere dichiarata `abstract` o `final`

E) La dichiarazione di un oggetto e la sua creazione possono essere svolte in tempi diversi

Versione 7 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) I modificatori che precedono una variabile di tipo array si applicano alla variabile array ed anche ai suoi elementi

B) Non è possibile dichiarare un attributo senza inizializzarlo

C) Una classe non interna può essere dichiarata `private`

D) Un attributo `static` non può essere acceduto mediante il nome della sua classe di appartenenza

E) La lunghezza di un array non può essere variata dopo la sua creazione

Versione 8 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) In Java non tutti i tipi numerici sono con segno

B) Prima dell'esecuzione del costruttore alle variabili di istanza sono assegnati i valori di default

C) Nell'overloading due metodi non possono avere lo stesso nome e lo stesso numero di parametri

D) La dichiarazione di un oggetto e la sua creazione possono essere svolte solo contemporaneamente

E) Un array vuoto può avere riferimento `null`

Versione 9 dell'esercizio 1

Dire quale delle seguenti affermazioni è vera:

A) Una classe non può essere dichiarata abstract o final

B) I modificatori applicati a una variabile di tipo array non si applicano alla variabile array ma ai suoi elementi

C) Due metodi non possono differire per il tipo di ritorno

D) La lunghezza di un array non può essere variata dopo la sua costruzione

E) Non è possibile dichiarare un attributo senza inizializzarlo

Versione 1 dell'esercizio 2

Date le dichiarazioni:

```
Exception a;
Error b;
Object n;
b = new Error();
a = new Exception();
n = new Exception();
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `a = (Exception) b;`

B) `b = (Error) n;`

C) `a = (Exception) n;`

D) `b = (Error) a;`

E) Nessuno dei precedenti

Versione 2 dell'esercizio 2

Date le dichiarazioni:

```
Exception e;
Integer m;
Object u;
u = new Exception();
e = new Exception();
m = new Integer(50);
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `m = (Integer) u;`

B) `e = (Exception) u;`

C) `e = (Exception) m;`

D) `m = (Integer) e;`

E) Nessuno dei precedenti

Versione 3 dell'esercizio 2

Date le dichiarazioni:

```
class Super extends Object {...}
class B1 extends Super {...}
class Sub1 extends Super {...}
```

e le inizializzazioni di variabile:

```
Super b;
B1 e;
Sub1 w;
e = new B1();
b = new B1();
w = new Sub1();
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `e = (B1) w;`

B) `e = (B1) b;`

C) `w = (Sub1) e;`

D) `w = (Sub1) b;`

E) Nessuno dei precedenti

Versione 4 dell'esercizio 2

Date le dichiarazioni:

```
Exception b;
Integer f;
Object t;
b = new Exception();
f = new Integer(5);
t = new Exception();
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `b = (Exception) t;`

B) `f = (Integer) b;`

C) `f = (Integer) t;`

D) `b = (Exception) f;`

E) Nessuno dei precedenti

Versione 5 dell'esercizio 2

Date le dichiarazioni:

```
String d;
Object m;
Error u;
u = new Error();
m = new Error();
d = new String("a");
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `d = (String) m;`

B) `u = (Error) d;`

C) `d = (String) u;`

D) `u = (Error) m;`

E) Nessuno dei precedenti

Versione 6 dell'esercizio 2

Date le dichiarazioni:

```
class Sub1 extends C0 {...}
class Sub2 extends C0 {...}
class C0 extends Object {...}
```

e le inizializzazioni di variabile:

```
Sub1 c;
Sub2 h;
C0 s;
c = new Sub1();
h = new Sub2();
s = new Sub2();
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `c = (Sub1) s;`

B) `h = (Sub2) s;`

C) `h = (Sub2) c;`

D) `c = (Sub1) h;`

E) Nessuno dei precedenti

Versione 7 dell'esercizio 2

Date le dichiarazioni:

```
Object [] e;
Object [] [] m;
Integer [] u;
u = new Integer [4];
m = new Object [3] [1];
e = new Object [8] [1];
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `m = (Object [] []) e;`

B) `m = (Object [] []) u;`

C) `u = (Integer []) e;`

D) `u = (Integer []) m;`

E) Nessuno dei precedenti

Versione 8 dell'esercizio 2

Date le dichiarazioni:

```
class B extends A {...}
class A extends Object {...}
class B1 extends A {...}
```

e le inizializzazioni di variabile:

```
A d;
B f;
B1 x;
f = new B();
x = new B1();
d = new B1();
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `f = (B) x;`

B) `x = (B1) f;`

C) `x = (B1) d;`

D) `f = (B) d;`

E) Nessuno dei precedenti

Versione 9 dell'esercizio 2

Date le dichiarazioni:

```
Object c;
Integer g;
Exception s;
g = new Integer(5);
s = new Exception();
c = new Integer(50);
```

indicare quale dei seguenti assegnamenti è corretto a tempo di esecuzione.

A) `s = (Exception) g;`

B) `g = (Integer) c;`

C) `s = (Exception) c;`

D) `g = (Integer) s;`

E) Nessuno dei precedenti

Versione 1 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) Quando viene invocata una macro (come le #define in C), viene inserito un record di attivazione nello stack?

   A) si
   B) no
   C) dipende

ii) Il controllo di correttezza dei downcast richiede controlli a runtime?

   A) si
   B) no
   C) dipende

Versione 2 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) Il primo garbage collector è stato introdotto

   A) in Java

   B) in C#

   C) in Lisp

ii) È vero che In un linguaggio funzionale puro, un identificatore x in una espressione rappresenta env(x)?

   A) si

   B) no

   C) dipende

Versione 3 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) I parametri IN passati per riferimento

   A) non possono essere letti prima di essere inizializzati

   B) non possono essere modificati

   C) nessuna delle precedenti

ii) Nei linguaggi con implementazione ibrida compilata/interpretata l'input del programma viene elaborato

   A) dal compilatore

   B) dall'interprete

Versione 4 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) I parametri OUT passati per copia

   A) non possono essere letti prima di essere inizializzati

   B) non possono essere modificati

   C) nessuna delle precedenti

ii) Nel primo Fortran l'occupazione di memoria di un programma era nota a compile time?

   A) si

   B) no

Versione 5 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) Il polimorfismo che permette più controlli a tempo di compilazione è quello

   A) per inclusione
   B) parametrico

ii) In Java l'ambiente non locale di una classe interna statica si può trovare

   A) sullo stack
   B) nello heap
   C) nella zona statica

Versione 6 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) Il Lisp supporta la ricorsione?

A) si

B) no

ii) L'aliasing consiste nel riferirsi alla stessa locazione con nomi diversi?

A) si

B) no

Versione 7 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) Il polimorfismo che permette più controlli a tempo di compilazione è quello

   A) per inclusione

   B) parametrico

ii) Il C supporta l'equivalenza per nome

   A) sui tipi primitivi

   B) sulle struct

Versione 8 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) È vero che In un linguaggio funzionale puro, un identificatore x in una espressione rappresenta env(x)?

   A) si

   B) no

   C) dipende

ii) Il primo Fortran aveva un garbage collector?

   A) si

   B) no

Versione 9 dell'esercizio 3

Indicare tutte le risposte vere, separando quelle di domande diverse con un punto e virgola.

i) In un programma che usa record varianti (come le union del C) il controllo dei tipi può essere fatto interamente a tempo di compilazione?

   A) si

   B) no

ii) Le macro (come ad es. le #define del C) hanno un proprio ambiente locale implementato con un record di attivazione?

   A) si

   B) no

   C) dipende

Versione 1 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*x = y[a]$, sapendo che x è un puntatore e che y è un vettore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $env(y) + mem(env(a))$.

Versione 2 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $x = \&y$.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(env(y) + 2)$.

Versione 3 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*x) = y[a]$, sapendo che x è un puntatore e che y è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a))$ e $mem(mem(env(y)))$.

Versione 4 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*p + z) = y$.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(env(y) + mem(mem(env(p))))$.

Versione 5 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(*x) = *a$, sapendo che x è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(a)) + 5$ e $mem(env(y))$.

Versione 6 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*x = y[2]$, sapendo che x è un puntatore e che y è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(env(y))$.

Versione 7 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*x = y[1]$, sapendo che x è un puntatore e che y è un puntatore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $env(x) + 1$ e $env(y) + mem(env(a))$.

Versione 8 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $*(a + z) = \&(y[1])$, sapendo che y è un vettore.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $mem(mem(env(y)))$.

Versione 9 dell'esercizio 4

i) Esprimere in termini di mem e env le parti sinistra e destra dell'assegnamento $x = \&y$.

ii) Scrivere un assegnamento le cui parti sinistra e destra corrispondono rispettivamente a $mem(env(x))$ e $env(y) + mem(env(a))$.

Versione 1 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends Exception { }
class MyExc3 extends MyExc2 { }
public class D1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         q();
         System.out.print(1);
      }
      catch( MyExc2 c ) {
         throw( new MyExc2() );
      }
      finally {
         System.out.print(2);
      }
   }
   static void q()
     throws Exception {
      try {
         throw( new MyExc2() );
      }
   }
}
```

A) Exception in thread "main" MyExc2

B) 2Exception in thread "main" MyExc2

C) Errore a tempo di compilazione

D) ... (ciclo infinito)

E) Nessuna delle precedenti

Versione 2 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends Exception { }
class MyExc3 extends MyExc2 { }
public class C1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         System.out.print(1);
         n();
         System.out.print(2);
      }
      catch( Exception s ) {
         System.out.print(3);
      }
      finally {
         System.out.print(4);
         throw( new Exception() );
      }
   }
   static void n()
     throws Exception {
      try {
         throw( new MyExc1() );
      }
      catch( Exception j ) {
         throw( new MyExc3() );
      }
      catch( MyExc2 c ) {
         System.out.print(5);
      }
      catch( MyExc1 d ) {
         System.out.print(6);
         throw( new MyExc2() );
      }
   }
}
```

A) Errore a tempo di compilazione

B) 1324Exception in thread "main" java.lang.Exception

C) 164Exception in thread "main" java.lang.Exception

D) 134Exception in thread "main" java.lang.Exception

E) Nessuna delle precedenti

Versione 3 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends MyExc1 { }
class MyExc3 extends MyExc2 { }
public class D1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         m();
      }
      catch( Exception r ) {
         System.out.print(1);
      }
      catch( MyExc1 s ) {
         System.out.print(2);
         throw( new MyExc3() );
      }
      finally {
         System.out.print(3);
      }
   }
   static void m()
     throws Exception {
      try {
         System.out.print(4);
         throw( new Exception() );
      }
      catch( MyExc1 c ) {
         System.out.print(5);
      }
      catch( Exception b ) {
         System.out.print(6);
         throw( new MyExc3() );
      }
      catch( MyExc2 e ) {
      }
      finally {
         System.out.print(7);
         throw( new MyExc3() );
      }
   }
}
```

A) Errore a tempo di compilazione

B) 46713

C) 465753

D) 463

E) Nessuna delle precedenti

Versione 4 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends MyExc1 { }
class MyExc3 extends MyExc2 { }
public class C1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         System.out.print(1);
         m();
      }
      catch( MyExc1 t ) {
         System.out.print(2);
      }
      catch( MyExc2 c ) {
         System.out.print(3);
      }
   }
   static void m()
     throws Exception {
      try {
         System.out.print(4);
         throw( new Exception() );
      }
      catch( Exception i ) {
         System.out.print(5);
         throw( new MyExc3() );
      }
      catch( MyExc3 t ) {
      }
      catch( MyExc1 u ) {
      }
   }
}
```

A) 145555555... (ciclo infinito)

B) 145Exception in thread "main" MyExc3

C) 1452

D) Errore a tempo di compilazione

E) Nessuna delle precedenti

Versione 5 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends MyExc1 { }
class MyExc3 extends MyExc2 { }
public class C1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         m();
      }
      catch( Exception a ) {
         System.out.print(1);
      }
      catch( MyExc1 u ) {
         System.out.print(2);
      }
      finally {
         throw( new MyExc3() );
      }
   }
   static void m()
     throws Exception {
      try {
         System.out.print(3);
         throw( new Exception() );
      }
      catch( MyExc2 t ) {
         System.out.print(4);
      }
      catch( MyExc1 v ) {
         System.out.print(5);
      }
      catch( Exception i ) {
         System.out.print(6);
      }
   }
}
```

A) 361

B) 36

C) Errore a tempo di compilazione

D) 36Exception in thread "main" MyExc3

E) Nessuna delle precedenti

Versione 6 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends Exception { }
class MyExc3 extends MyExc2 { }
public class C1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         System.out.print(1);
         n();
         System.out.print(2);
      }
      catch( Exception d ) {
         System.out.print(3);
      }
      catch( MyExc3 r ) {
         System.out.print(4);
      }
      finally {
         System.out.print(5);
      }
   }
   static void n()
     throws Exception {
      try {
         System.out.print(6);
         throw( new MyExc1() );
      }
      catch( Exception j ) {
      }
      catch( MyExc2 z ) {
         throw( new MyExc2() );
      }
      catch( MyExc3 v ) {
      }
      finally {
         System.out.print(7);
      }
   }
}
```

A) 1672

B) Errore a tempo di compilazione

C) 16725

D) 1675Exception in thread "main" MyExc1

E) Nessuna delle precedenti

Versione 7 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends Exception { }
class MyExc3 extends MyExc2 { }
public class B1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         n();
      }
      catch( MyExc1 z ) {
      }
      catch( Exception g ) {
         System.out.print(1);
      }
      catch( MyExc2 g ) {
      }
   }
   static void n()
     throws Exception {
      try {
         throw( new MyExc3() );
      }
      catch( MyExc2 h ) {
         System.out.print(2);
      }
      catch( MyExc3 h ) {
         System.out.print(3);
      }
      catch( MyExc1 k ) {
      }
      finally {
         System.out.print(4);
      }
   }
}
```

A) 24

B) 3

C) Errore a tempo di compilazione

D) 34

E) Nessuna delle precedenti

Versione 8 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends MyExc1 { }
class MyExc3 extends MyExc1 { }
public class C1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         System.out.print(1);
         q();
         System.out.print(2);
      }
      catch( MyExc2 y ) {
         throw( new Exception() );
      }
      catch( Exception x ) {
      }
      finally {
         System.out.print(3);
      }
   }
   static void q()
     throws Exception {
      try {
         throw( new MyExc1() );
      }
      catch( MyExc1 b ) {
         System.out.print(4);
      }
      catch( Exception s ) {
      }
      catch( MyExc2 t ) {
      }
      finally {
         throw( new MyExc2() );
      }
   }
}
```

A) 143Exception in thread "main" java.lang.Exception

B) 142

C) 14423

D) Errore a tempo di compilazione

E) Nessuna delle precedenti

Versione 9 dell'esercizio 5

Qual è l'output di questo codice?

```
class MyExc1 extends Exception { }
class MyExc2 extends Exception { }
class MyExc3 extends Exception { }
public class A1 {
   public static void main(String [] argv)
     throws Exception {
      try {
         System.out.print(1);
         m();
      }
      catch( Exception a ) {
         System.out.print(2);
      }
      catch( MyExc3 a ) {
         throw( new MyExc1() );
      }
      catch( MyExc2 z ) {
      }
   }
   static void m()
     throws Exception {
      try {
         throw( new MyExc1() );
      }
      catch( Exception x ) {
         System.out.print(3);
      }
      finally {
         System.out.print(4);
         throw( new MyExc2() );
      }
   }
}
```

A) 1342

B) 1343

C) Errore a tempo di compilazione

D) 14

E) Nessuna delle precedenti