**Garbage Collections**

**Q: 01 Given:**  
**1. public class GC {**  
**2. private Object o;**  
**3. private void doSomethingElse(Object obj) { o = obj; }**  
**4. public void doSomething() {**  
**5. Object o = new Object();**  
**6. doSomethingElse(o);**  
**7. o = new Object();**  
**8. doSomethingElse(null);**  
**9. o = null;**  
**10. }**  
**11. }**  
**When the doSomething method is called, after which line does the Object created in line 5 become**  
**available for garbage collection?**  
A. Line 5  
B. Line 6  
C. Line 7  
D. Line 8  
E. Line 9  
F. Line 10  
  
  
**Q: 02 Given:**  
**11. public void genNumbers() {**  
**12. ArrayList numbers = new ArrayList();**  
**13. for (int i=0; i<10; i++) {**  
**14. int value = i \* ((int) Math.random());**  
**15. Integer intObj = new Integer(value);**  
**16. numbers.add(intObj);**  
**17. }**  
**18. System.out.println(numbers);**  
**19. }**  
**Which line of code marks the earliest point that an object referenced by intObj becomes a candidate for**  
**garbage collection?**  
A. Line 16  
B. Line 17  
C. Line 18  
D. Line 19  
E. The object is NOT a candidate for garbage collection.  
  
  
**Q: 03 Given:**  
**11. rbo = new ReallyBigObject();**  
**12. // more code here**  
**13. rbo = null;**  
**14. /\* insert code here \*/**  
**Which statement should be placed at line 14 to suggest that the virtual machine expend effort toward**  
**recycling the memory used by the object rbo?**  
A. System.gc();  
B. Runtime.gc();  
C. System.freeMemory();  
D. Runtime.getRuntime().growHeap();  
E. Runtime.getRuntime().freeMemory();  
  
  
**Question: 05**  
**Which two are true? (Choose two.)**  
A. A finalizer may NOT be invoked explicitly.  
B. The finalize method declared in class Object takes no action.  
C. super.finalize() is called implicitly by any overriding finalize method.  
D. The finalize method for a given object will be called no more than  
once by the garbage collector.  
E. The order in which finalize will be called on two objects is based on  
the order in which the two objects became finalizable.  
  
  
**06. Given:**  
**class CardBoard {**  
**Short story = 5;**  
**CardBoard go(CardBoard cb) {**  
**cb = null;**  
**return cb;**  
**}**  
**public static void main(String[] args) {**  
**CardBoard c1 = new CardBoard();**  
**CardBoard c2 = new CardBoard();**  
**CardBoard c3 = c1.go(c2);**  
**c1 = null;**  
**// do Stuff**  
**} }**  
**When // doStuff is reached, how many objects are eligible for GC?**  
A. 0  
B. 1  
C. 2  
D. Compilation fails.  
E. It is not possible to know.  
F. An exception is thrown at runtime.

**07. Which is true? (Choose all that apply.)**  
A. The invocation of an object’s finalize() method is always the last thing that happens before an object is garbage collected (GCed).  
B. When a stack variable goes out of scope it is eligible for GC.  
C. Some reference variables live on the stack, and some live on the heap.  
D. Only objects that have no reference variables referring to them can be eligible for GC.  
E. It’s possible to request the GC via methods in either java.lang.Runtime or  
java.lang.System classes.

**08. Given:**  
**1. class Eco {**  
**2. public static void main(String[] args) {**  
**3. Eco e1 = new Eco();**  
**4. Eco e2 = new Eco();**  
**5. Eco e3 = new Eco();**  
**6. e3.e = e2;**  
**7. e1.e = e3;**  
**8. e2 = null;**  
**9. e3 = null;**  
**10. e2.e = e1;**  
**11. e1 = null;**  
**12. }**  
**13. Eco e;**  
**14. }**  
**At what point is only a single object eligible for GC?**  
A. After line 8 runs.  
B. After line 9 runs.  
C. After line 10 runs.  
D. After line 11 runs.  
E. Compilation fails.  
F. Never in this program.  
G. An exception is thrown at runtime.