1 Which statements about the `hashCode()` and `equals()` methods are true?

Select the **two** correct answers.

(a) Two objects that are different according to the `equals()` method, must have different hash values.

(b) Two objects that are equal according to the `equals()` method, must have the same hash value.

(c) Two objects that have the same hash value, must be equal according to the `equals()` method.

(d) Two objects that have different hash values, must be unequal according to the `equals()` method.

2 Given that the objects referenced by the parameters override the `equals()` and the `hashCode()` methods appropriately, which return values are possible from the following method?

```
String func(Object x, Object y) {

return (x == y) + " " + x.equals(y) + " " + (x.hashCode() == y.hashCode());

}
```

Select the **four** correct answers.

[✓ (a) "false false false"]

[✓ (b) "false false true"]

(c) "false true false"

[✓ (d) "false true true"]

(e) "true false false"

(f) "true false true"

(g) "true true false"

[✓ (h) "true true true"]

3 Which code, when inserted at (1), in the `equalsImpl()` method will provide a correct implementation of the `equals()` method?

```
public class Pair {
int a, b;
public Pair(int a, int b) {
this.a = a;
this.b = b;
}
public boolean equals(Object o) {
return (this == o) || (o instanceof Pair) && equalsImpl((Pair) o);
}
private boolean equalsImpl(Pair o) {
// (1) INSERT CODE HERE ...
}
}
```

Select the **three** correct answers.

[✓ (a) return a == o.a || b == o.b;]

(b) return false;

(c) return a >= o.a;

[✓ (d) return a == o.a;]

[✓ (e) return a == o.a && b == o.b;]

4 Which code, when inserted at (1), will provide a correct implementation of the

hashCode() method in the following program?

```
import java.util.*;
public class Measurement {
int count;
int accumulated;
public Measurement() {}
public void record(int v) {
count++;
accumulated += v;
}
public int average() {
return accumulated/count;
}
public boolean equals(Object other) {
if (this == other)
return true;
if (!(other instanceof Measurement))
return false;
Measurement o = (Measurement) other;
if (count != 0 && o.count != 0)
return average() == o.average();
return count == o.count;
}
public int hashCode() {    // (1) INSERT CODE HERE ...
} }
```

Select the **two** correct answers.

[✓ (a) return 31337;]

(b) return accumulated / count;

(c) return (count << 16) ^ accumulated;

[✓ (d) return ~accumulated;]

(e) return count == 0 ? 0 : average();

5 What will be the result of compiling and running the following program?

```
import java.util.Comparator;
class Person implements Comparable<Person> {
String name;
int age;
Person(String name, int age)
{ this.name = name; this.age = age; }

public int compareTo(Person p2) {
Comparator<String> strCmp = Person.cmp();
int status = strCmp.compare(this.name, p2.name);
if (status == 0) {
Comparator<Integer> intCmp = Person.cmp();
status = intCmp.compare(this.age, p2.age);
}
return status;
}
public static <E extends Comparable<E>>
Comparator<E> cmp() {
return new Comparator<E>() {
public int compare(E s1, E s2) { return
s2.compareTo(s1); }
};
}
public static void main(String[] args) {
Person p1 = new Person("Tom", 20);
Person p2 = new Person("Dick", 30);
Person p3 = new Person("Tom", 40);
System.out.println((p1.compareTo(p2) < 0) + " " +
(p1.compareTo(p3) < 0));
}
}
```

Select the one correct answer.

(a) The program will fail to compile.

(b) The program will compile but throw an exception when run.

(c) [✓ The program will compile and print `true false`, when run.]

(d) The program will compile and print `true true`, when run.

(e) The program will compile and print `false false`, when run.

(f) The program will compile and print `false true`, when run.

6 Which of these are core interfaces in the collections framework?

Select the three correct answers.

(a) `Set<E>`

(b) `Bag<E>`

(c) `LinkedList<E>`

(d) `Collection<E>`

(e) `Map<K,V>`

7 Which of these implementations are provided by the `java.util` package?

Select the two correct answers.

(a) `HashList<E>`

(b) `HashMap<K,V>`

(c) `ArraySet<E>`

(d) `ArrayMap<K,V>`

(e) `TreeMap<K,V>`

8 What is the name of the interface used to represent collections that maintain non-unique

Elements in order?

Select the one correct answer.

(a) Collection<E>

(b) Set<E>

(c) SortedSet<E>

(d) List<E>

(e) Sequence<E>

9 Which methods are specified by the `Iterator<E>` interface?

Select the three correct answers.

[✓ (a) `hasNext()`]

(b) `hasMore()`

[✓ (c) `remove()`]

(d) `delete()`

(e) `more()`

[✓ (f) `next()`]

10 Which identifiers, when inserted in appropriate places in the program, will result in the output 911?

```
Collection<_________> myItems = new ArrayList<___________>();

myItems.add(9); myItems.add(1); myItems.add(1);

Iterator<__________> iterator = ______________.iterator();

while (________________.____________()) {

System.out.print(______________._________());

}
```

Select the five correct answers.

(a) hasNext

(b) myItems

(c) next

(d) Integer

(e) int

(f) Collection

(g) iterator

11 What will the program print when it is compiled and run?

```
import java.util.ArrayList;

import java.util.Collection;

public class RQ400_100 {

public static void main(String[] args) {

int sum = 0;

for (int i : makeCollection())

sum += i;

System.out.println(sum);

}

static Collection<Integer> makeCollection() {

System.out.println("A collection coming up.");

Collection<Integer> collection = new

ArrayList<Integer>();

collection.add(10); collection.add(20);

collection.add(30);

return collection;

}

}
```

Select the one correct answer.

(a) A collection coming up.

60

(b) A collection coming up.

A collection coming up.

A collection coming up.

60

(c) The program does not compile.

(d) None of the above.

12 Which statements are true about the `for(:)` loop:

`for (`*type variable* `:` *expression*`)` *statement*

Select the **three** correct answers.

(a) The *variable* is only visible in the `for(:)` loop body.

(b) The *expression* is only evaluated once.

(c) The type of the expression must be `java.lang.Iterable` or an array type.

(d) Changing the value of the *variable* in the loop body affects the data structure represented by the *expression*.

(e) The loop runs backwards if the *expression* is negated as follows: `!`*expression*.

(f) We can iterate over several data structures simultaneously in a `for(:)` loop.

13 What will the program print when compiled and run?

```
import java.util.ArrayList;

import java.util.Collection;

public class IterateOverCollection2 {

public static void main(String[] args) {

Collection<String> words = new ArrayList<String>();


words.add("Don't"); words.add("change");

words.add("me!");

System.out.println("Before: " + words);

for (String word : words) {

System.out.print(word.toUpperCase() + "_");

}

System.out.println();

System.out.println("After: " + words);

}

}
```

Select the one correct answer.

(a) Before: [Don't, change, me!]

DON'T_CHANGE_ME!_

After: [DON'T, CHANGE, ME!]

[✓ (b)] Before: [Don't, change, me!]

DON'T_CHANGE_ME!_

After: [Don't, change, me!]

(c) The program will throw a java.util.ConcurrentModificationException, when

run.

(d) The program fails to compile.

14 Which code, when inserted at (1), will result in the following output:

```
Before: [Apple, Orange, Apple]

After: [Orange]
```

from the program when compiled and run?

```
import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

class Fruity {

private String fName;

Fruity(String fName) { this.fName = fName; }

public void setName(String newName) { this.fName =

newName; }

public String toString() { return fName; }

public boolean equals(Object other) {

if (this == other) return true;

if (!(other instanceof Fruity)) return false;

return

fName.equalsIgnoreCase(((Fruity)other).fName);

} }

public class RQ400_50 {

public static void main(String[] args) {

Fruity apple = new Fruity("Apple");

Fruity orange = new Fruity("Orange");

List<Fruity> list = new ArrayList<Fruity>();

list.add(apple); list.add(orange); list.add(apple);

System.out.println("Before: " + list);

// (1) INSERT CODE HERE ...

System.out.println("After: " + list);

} }
```

Select the two correct answers.

(a)
```
for (Fruity f : list) {

if (f.equals(apple))

list.remove(f);

}
```

(b)
```
int i = 0;

for (Fruity f : list) {

if (f.equals(apple))

list.remove(i);

i++;

}
```

(c)
```
for (int j = 0; j < list.size(); j++) {

Fruity f = list.get(j);

if (f.equals(apple))

list.remove(j);

}
```

(d)
```
Iterator<Fruity> itr = list.iterator();

while (itr.hasNext()) {

Fruity f = itr.next();

if (f.equals(apple))

itr.remove(); }
```

15 Which statements about collections are true?

Select the two correct answers.

(a) Some operations on a collection may throw an UnsupportedOperationException.

(b) Methods calling optional operations in a collection must either catch an

UnsupportedOperationException or declare it in their throws clause.

(c) A List can have duplicate elements.

(d) An ArrayList can only accommodate a fixed number of elements.

(e) The Collection interface contains a method named get.