Question 1.

Given:
```
public class Book{
      private int id;
      private String title;
      //constructors and accessors not shown
}
```

Assuming that book is a reference to a valid Book object, which of the following code fragments correctly prints the details of the Book?

Select 1 option:

A.
```
Consumer<Book> c = b->b.getId()+":"+b.getTitle();
c.accept(book);
```

B.
```
Consumer<Book> c = b->System.out.println(b.getId()+":"+b.getTitle());
c.accept(book);
```

C.
```
Consumer<Book> c = b->{ String details = b.getId()+":"+b.getTitle();};
c.accept(book);
System.out.println(c);
```

D.
```
Consumer<Book> c = System.out::println;
c.accept(book);
```

Question 2.

What will the following code print when compiled and run?

```
public class TestClass{

      public static int operate(IntUnaryOperator iuo){
            return iuo.applyAsInt(5);
      }

      public static void main(String[] args) {

            IntFunction<IntUnaryOperator> fo = a->b->a-b; //1

            int x = operate(fo.apply(20)); //2
            System.out.println(x);
      }
}
```

Select 1 option:

A. Compilation error at //1

B. Compilation error at //2

C. 15

D. -15

E. 20

F. Exception at run time

Question 3.

Which of the following interface definitions can be implemented using Lambda expressions?

```
interface A{
      static void m1(){};
}
interface AA extends A{
      void m2();
}
interface AAA extends AA{
      void m3();
}
interface B {
      default void m1(){ }
}
interface BB extends B {
      static void m2(){ };
}
interface C extends BB{
      void m3();
}
```

Select 2 options:

A. A

B. AA

C. AAA

D. B

E. BB

F. C

Question 4.

Which method should also be overridden if public boolean equals(Object obj) is being overridden in a class?

Select 1 option:

A. `public int hashCode(Object obj);`

B. `public int hashCode();`

C. `public Object clone();`

D. `public int compareTo(Object obj);`

E. `public void finalize();`

Question 5.

Consider the following class:

```
public class IntPair {
      private int a;
      private int b;
      public void setA(int i){ this.a = i; }
      public int getA(){ return this.a; }
      public void setB(int i){ this.b = i; }
      public int getB(int b){ return b; }
      public boolean equals(Object obj) {
            return ( obj instanceof IntPair && this.a == ((IntPair) obj).a && this.b ==
                  ((IntPair) obj).b );
      }
      public int hashCode() {
            //1
      }
}
```

Which of the following options would be valid at **//1**?

Select 1 option:

A. `return a;`

B. `return a*b;`

C. `return a+b;`

D. `return b;`

E. All of the above are valid.

Question 6.

The `hashCode()` method of a class can be used to test inequality but not equality of two objects of that class.
In other words, if the hash codes of two objects of the same class are not equal, you can definitely say that the objects are not equal, However, if the hash codes are equal, you cannot say anything about the equality of the objects.

(Assume that the equals and hashCode contract is not violated by the class.)

Select 1 option:

A. True

B. False

Question 7.

Given the following definition of class, which member variables are accessible from OUTSIDE the package `com.enthu.qb`?

```
package com.enthu.qb;
public class TestClass {
      int i;
      public int j;
      protected int k;
      private int l;
}
```

Select 2 options:

A. Member variable i.

B. Member variable j.

C. Member variable k.

D. Member variable k, but only for subclasses.

E. Member variable l.

Question 8.

Consider the following code:

```
class TaskBase {
      int getStatusCode(Object obj) throws NullPointerException {
            if(obj != null ) return 1;
            else return 0;
      }
}
class ParallelTask extends TaskBase {
      //override getStatusCode method.
}
```

Which of the following statements are valid?

Select 2 options:

A. Overriding method can take String as a parameter.

B. Overriding method can return a long instead of int.

C. Overriding method can throw any RuntimeException.

D. Overriding method cannot throw any checked exception.

E. Overriding method may declare the return type as byte.

Question 9.

Carefully examine the following code.

```
public class StaticTest {
      static
      {
            System.out.println("In static");
      }
      {
            System.out.println("In non - static");
      }
      public static void main(String args[ ]) {
            StaticTest st1;                    //1
            System.out.println(" 1 ");
            st1 = new StaticTest();            //2
            System.out.println(" 2 ");
            StaticTest st2 = new StaticTest(); //3
      }
}
```

What will be the output?

Select 1 option:

A. In static, 1, In non - static, 2, In non - static **:in that order.**

B. Compilation error.

C. 1, In static, In non - static, 2, In non - static **: in that order.**

D. In static, 1, In non - static, 2, In non - static **: in unknown order.**

Question 10.

Which statements about the following code are correct?

```
interface House{
      public default String getAddress(){
            return "101 Main Str";
      }
}

interface Office {
      public static String getAddress(){
            return "101 Smart Str";
      }
}

interface WFH extends House, Office{
}

class HomeOffice implements House, Office{
      public String getAddress(){
            return "R No 1, Home";
      }
}

public class TestClass {

      public static void main(String[] args) {
            Office off = new HomeOffice();            //1
            System.out.println(off.getAddress());     //2
      }
}
```

Select 1 option:

A. Code for class HomeOffice will cause compilation to fail.

B. Code for interface WFH will cause compilation to fail.

C. It will compile fine and print R No 1, Home when run.

D. Line at //1 will cause compilation to fail.

E. Line at //2 will cause compilation to fail.

Question 11.

Identify the correct statements about the following code:

```
import java.util.*;
class Person {
      private static int count = 0;
      private String id = "0"; private String interest;
      public Person(String interest){
            this.interest = interest; this.id = "" + ++count;
      }
      public String getInterest(){ return interest; }
      public void setInterest(String interest){ this.interest = interest; }
      public String toString(){ return id; }
}

public class StudyGroup {
      String name = "MATH";
      TreeSet<Person> set = new TreeSet<Person>();
      public void add(Person p) {
            if(name.equals(p.getInterest())) set.add(p);
      }

      public static void main(String[] args) {
            StudyGroup mathGroup = new StudyGroup();
            mathGroup.add(new Person("MATH"));
            System.out.println("A");
            mathGroup.add(new Person("MATH"));
            System.out.println("B");
            System.out.println(mathGroup.set);
      }
}
```

Select 1 option:

A. It will print : A, B, and then the contents of mathGroup.set.

B. It will compile with a warning.

C. It will NOT throw an exception at runtime.

D. It will compile without warning but will throw an exception at runtime.

E. It will only print : A

F. It will print : A and B.

Question 12.

What will be the contents of d at the end of the following code?

```
      Deque<Integer> d = new ArrayDeque<>();
      d.add(1);
      d.push(2);
      d.pop();
      d.offerFirst(3);
      d.remove();
      System.out.println(d);
```

Select 1 option:

A. 1

B. 2

C. 3

D. Exception at run time.

E. It will not compile.

Question 13.

Consider the following code:

//Assume appropriate imports
```
class Person{
      String name;
      String dob;
      public Person(String name, String dob){
            this.name = name; this.dob = dob;
      }
}
public class SortTest {
      public static void main(String[] args) {
            ArrayList<Person> al = new ArrayList<>();
            al.add(new Person("Paul", "01012000"));
            al.add(new Person("Peter", "01011990"));
            al.add(new Person("Patrick", "01012002"));

            //INSERT CODE HERE

            for(Person a : al) System.out.println(a.name+" "+ a.dob);
      }
}
```

What can be inserted in the code so that it will sort the collection of Persons by Person's dob attribute?

Select 1 option:

A.
```
Collections.sort(al, new Comparable<Person>(){
    public int compare(Person o1, Person o2) {
           return o1.dob.compareTo(o2.dob);
    }
});
```

B.
```
Collections.sort(al, new Comparator<Person>(){
    public int compare(Person o1, Person o2) {
           return o1.dob.compareTo(o2.dob);
    }
});
```

C.
```
Collections.sort(al, new Comparable<Person>(){
    public int compare(Person o1, Person o2) {
           return o1.dob.compare(o2.dob);
    }
});
```

D.
```
Collections.sort(al, new Comparator<Person>(){
    public int compare(Person o1, Person o2) {
           return o1.dob.compare(o2.dob);
    }
});
```

Question 14.

Given that the user account under which the following code is run does not have the permission to access a.java, what will the the following code print when run?

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.nio.file.AccessDeniedException;
import java.nio.file.NoSuchFileException;

public class IOTest {
      public static void main(String[] args) {
            try(BufferedReader bfr = new BufferedReader(new
                              FileReader("c:\\works\\a.java"))){
                  String line = null;
                  while( (line = bfr.readLine()) != null){
                        System.out.println(line);
                  }
            }catch(NoSuchFileException|IOException|AccessDeniedException e){
                  e.printStackTrace();
            }
      }
}
```

Select 1 option:

A. It will run without any exception and without any output.

B. It will print a stack trace for AccessDeniedException.

C. It will print a stack trace for IOException.

D. It will print a stack trace for NoSuchFileException.

E. It will not compile because the catch clause is invalid.

Question 15.

Given:

```
class Game{ }
class Cricket extends Game{ }
class Instrument{ }
class Guitar extends Instrument{ }

interface Player<E>{ void play(E e); }
interface GamePlayer<E extends Game> extends Player<E>{ }
interface MusicPlayer<E extends Instrument> extends Player{ }
```

Identify valid declarations.

Select 1 option:

A.
```
class Batsman implements GamePlayer<Cricket>{
    public void play(Game o){ }
}
```

B.
```
class Bowler implements GamePlayer<Guitar>{
    public void play(Guitar o){ }
}
```

C.
```
class Bowler implements Player<Guitar>{
    public void play(Guitar o){ }
}
```

D.
```
class MidiPlayer implements MusicPlayer {
    public void play(Guitar g){ }
}
```

E.
```
class MidiPlayer implements MusicPlayer<Instrument> {
    public void play(Guitar g){ }
}
```

Question 16.

Which of the following are characteristics of a class that implements the Singleton pattern?

Select 2 options:

A. The class is private.

B. The class has no constructor.

C. The class has a public static method that returns an instance of that class.

D. The class has a private class variable that refers to an instance of the same class.

E. The constructor of the class resolves to the same object each time it is invoked.

F. The class implements Singleton interface.

Question 17.

What can you do to make a class immutable?

NOTE: This statement is ambiguous. The immutability could refer to the immutability of two things - class or the objects of the class. Expect such ambiguity in the real exam questions as well. Our guess is that it refers to the objects of a class.

Select 3 options:

A. Make constructor private.

B. Make member fields private and final and include public getters for them.

C. Make the class final.

D. Instead of a default no-args constructor, create a constructor that initializes the members.

E. Declare the members protected.

F. Declare members static.

G. Make the setters private.

Question 18.

Given:

```
ArrayList<Integer> source = new ArrayList<Integer>();
source.addAll(Arrays.asList(1, 2, 3, 4, 5, 6));

List<Integer> destination =
      Collections.synchronizedList(new ArrayList<Integer>());

source
      .parallelStream()                             //1
      .peek(item->{destination.add(item); })        //2
      .forEachOrdered(System.out::print);

System.out.println("");

destination
      .stream()                                     //3
      .forEach(System.out::print);                  //4

System.out.println("");
```

What changes must be made to the above code so that it will consistently print

```
123456
123456
?
```

Select 1 option:

A. Replace code at //1 with
`.stream()`

B. Replace code at //2 with
`.map(item->{destination.add(item); return item; })`

C. Replace code at //3 with
`.parallelStream()`

D. Replace code at //4 with
`.forEachOrdered(System.out::print);`

Question 19.

Given:

```
import java.util.Iterator;
import java.util.Map.Entry;
import java.util.concurrent.ConcurrentHashMap;

public class Cache {

      static ConcurrentHashMap<String, Object> chm =
            new ConcurrentHashMap<String, Object>();

      public static void main(String[] args) {
            chm.put("a", "aaa");
            chm.put("b", "bbb");
            chm.put("c", "ccc");

            new Thread(){
                  public void run(){
                        Iterator<Entry<String, Object>> it =
                              Cache.chm.entrySet().iterator();
                        while(it.hasNext()){
                              Entry<String, Object> en = it.next();
                              if(en.getKey().equals("a") || en.getKey().equals("b")){
                                    it.remove();
                              }
                        }
                  }
            }.start();

            new Thread(){
                  public void run(){
                        Iterator<Entry<String, Object>> it =
                              Cache.chm.entrySet().iterator();
                        while(it.hasNext()){
                              Entry<String, Object> en = it.next();
                              System.out.print(en.getKey()+", ");
                        }
                  }
            }.start();
      }
}
```

Which of the following are possible outputs when the above program is run?

Select 1 option:

A. It may print any combination of the keys.

B. It may print any combination except: c,

C. It may print any combination except: a, or b, or a, b, or b, a

D. It may print any combination except: b, c,

E. It may print any combination except: a, b,

Question 20.

A programmer is writing a small component that processes a file line by line. The following is the code :

//import statements not shown
```
public class LineByLineProcessor {

      public void processLines(String fullFilePath) throws Exception {
            // declare and initialize "handle" here
            String str = null;
            while( (str = handle.readLine()) != null) {
                  System.out.println("Processing line : "+str);
            }
            handle.close();
      }
}
```

Which of the given options will declare and initialize handle appropriately?

Select 2 options:

A. `Reader handle = new FileReader(fullFilePath);`

B. `BufferedReader handle = new BufferedReader(fullFilePath);`

C. `BufferedReader handle = new BufferedReader(new File(fullFilePath));`

D. `BufferedReader handle = new BufferedReader(new FileReader(fullFilePath));`

E. `BufferedReader handle = new BufferedReader(new FileReader( new File(fullFilePath)));`

Question 21.

Given:

```
public static void createFile(String name) throws Exception{
      try (
            OutputStream os = new FileOutputStream(name); ) {

            //INSERT CODE HERE

            //flush and close the streams that are open
      }
}
```

Which of the following combinations of the lines of code and their outcome when inserted above, are correct?

Select 2 options:

A.
```
PrintWriter pw = new PrintWriter(os);
pw.write(1);
```
Size of the file depends on default character encoding.

B.
```
os.write(99);
```
A file of size 1 byte will be created.

C.
```
BufferedOutputStream bos = new BufferedOutputStream(os);
PrintWriter pw = new PrintWriter(bos);
pw.print(99);
```
A file of size 1 byte will be created.

D.
```
os.writeInt(99);
```
A file of size 4 bytes will be created.

E.
```
PrintWriter pw = new PrintWriter(os);
pw.writeInt(1);
```
A file of size 4 bytes will be created.

Question 22.

Consider following two classes:

//in file A.java
```
package p1;
public class A {
      protected int i = 10;
      public int getI() { return i; }
}
```

//in file B.java
```
package p2;
import p1.*;
public class B extends p1.A {
      public void process(A a) {
            a.i = a.i*2;
      }
      public static void main(String[] args) {
            A a = new B();
            B b = new B();
            b.process(a);
            System.out.println( a.getI() );
      }
}
```

What will be the output of compiling and running class B ?

Select 1 option:

A. It will print 10.

B. It will print 20.

C. It will not compile.

D. It will throw an exception at Run time.

E. None of the above.

Question 23.

Consider the following code:

```
class A {
      A() {  print();   }
      private void print() { System.out.println("A"); }
}

class B extends A {
      int i =   Math.round(3.5f);
      public static void main(String[] args) {
            A a = new B();
            a.print();
      }
      void print() { System.out.println(i); }
}
```

What will be the output when class B is run ?

Select 1 option:

A. It will print A, 4.

B. It will print A, A

C. It will print 0, 4

D. It will print 4, 4

E. None of the above.

Question 24.

Expression (s instanceof java.util.Date) will return false if 's' was declared as a variable of class java.lang.String.

Select 1 option:

A. True

B. False

Question 25.

Consider the following code:

```
public class TestOuter {
      public void myOuterMethod() {
            // 1
      }
      public class TestInner { }
      public static void main(String[] args) {
            TestOuter to = new TestOuter();
            // 2
      }
}
```

Which of the following options correctly instantiates a TestInner object?

Select 2 options:

A. `new TestInner();` at //1

B. `new TestInner();` at //2

C. `new to.TestInner();` at //2

D. `new TestOuter.TestInner();` at //2

E. `new TestOuter().new TestInner();` at //1

Question 26.

Identify the correct statement(s).

Select 2 options:

A. To customize the behavior of class serialization, the readObject and writeObject methods should be overridden.

B. To customize the behavior of class serialization, the defaultReadObject and defaultWriteObject methods should be overridden.

C. Objects of a final class cannot be serialized.

D. Constructor of the class for an object being deserialized is never invoked.

E. While serializing an object, only those objects in the object graph that implement Serializable are serialized and the rest are ignored.

Question 27.

Consider the following code:

```
class Bond
{
      String ticker; double coupon; java.util.Date maturity;
}

class Portfolio implements Serializable
{
      String accountName;
      Bond[] bonds;
}

public class TestClass {
      public static void main(String[] args) throws Exception{
            Portfolio portfolio = // get portfolio somehow.
            // serialize portfolio
      }
}
```

Which of the following approaches can be taken independent of each other so that a Portfolio object can be serialized while preserving the state of the Bond objects contained in Portfolio?

Select 2 options:

A. It can be serialized as it is without any modification.

B. Just have Bond class implement Serializable.

C. Just make 'bonds' field in Portfolio transient.

D. Change the type of bonds from Bond[] to ArrayList<Bond> bonds;

E. Make bonds array transient in Portfolio and implement readObject(ObjectOutputStream os) and writeObject(ObjectOutputStream os) methods to read and write the state of Bond objects explicitly.

Question 28.

Given:

```
public class Item{
      private String name;
      private String category;
      private double price;

      public Item(String name, String category, double price){
            this.name = name;
            this.category = category;
            this.price = price;
      }

      //accessors not shown
}
```

What will the following code print?

```
List<Item> items = Arrays.asList(
      new Item("Pen", "Stationery", 3.0),
      new Item("Pencil", "Stationery", 2.0),
      new Item("Eraser", "Stationery", 1.0),
      new Item("Milk", "Food", 2.0),
      new Item("Eggs", "Food", 3.0)
);

ToDoubleFunction<Item> priceF = Item::getPrice; //1

items.stream()
      .collect(Collectors.groupingBy(Item::getCategory)) //2
      .forEach((a, b)->{
            double av = b.stream().collect(Collectors.averagingDouble(priceF)); //3
            System.out.println(a+" : "+av);
      });
```

Select 1 option:

A. Stationery : 2.0
Food : 2.5

B. [Pen : 3.0, Pencil : 2.0, Eraser : 1.0] : 2.0
[Milk : 2.0, Eggs : 3.0] : 2.5

C. Pen : 3.0, Pencil : 2.0, Eraser : 1.0 : 2.0
Milk : 2.0, Eggs : 3.0 : 2.5

D. Stationery : [Pen : 3.0, Pencil : 2.0, Eraser : 1.0] : 2.0
Food : [Milk : 2.0, Eggs : 3.0] : 2.5

E. Compilation error due to code at //1

F. Compilation error due to code at //2

G. Compilation error due to code at //3

Question 29.

What will the following code print when compiled and run?

```
Stream<Integer> values = IntStream.rangeClosed(10, 15).boxed();        //1
Object obj = values.collect(Collectors.partitioningBy(x->x%2==0));     //2
System.out.println(obj);
```

Select 1 option:

A. Compilation error at //1

B. Compilation error at //2

C. {[11, 13, 15], [10, 12, 14]}

D. [[11, 13, 15], [10, 12, 14]]

E. {false=[11, 13, 15], true=[10, 12, 14]}

Question 30.

Which of the following standard functional interfaces is most suitable to process a large collection of int primitives and return processed data for each of them?

Select 1 option:

A. `Function<Integer>`

B. `IntFunction`

C. `Consumer<Integer>`

D. `IntConsumer`

E. `Predicate<Integer>`

Question 31.

Given that `java.lang.String` has two overloaded `toUpperCase` methods - `toUpperCase()` and `toUpperCase(Locale)`, consider the following code:

```
String name = "bob";
String val = null;
//Insert code here
System.out.print(val);
```

Which of the following code fragments can be inserted in the above code so that it will print BOB?

Select 2 options:

A.
```
Supplier<String> s = name::toUpperCase;
val = s.get();
```

B.
```
Supplier<String> s = name::toUpperCase;
val = s.apply();
```

C.
```
Function<String> f = name::toUpperCase;
val = f.get();
```

D.
```
Function<String> f = name::toUpperCase;
val = f.apply();
```

E.
```
Function<String, Locale> f = name::toUpperCase;
val = f.apply();
```

F. Only one of the above is correct.

Question 32.

Which statements about the following code are correct?

```
interface House{
      public default String getAddress(){
            return "101 Main Str";
      }
}

interface Office {
      public default String getAddress(){
            return "101 Smart Str";
      }
}

class HomeOffice implements House, Office{
      public String getAddress(){
            return "R No 1, Home";
      }
}

public class TestClass {

      public static void main(String[] args) {
            House h = new HomeOffice();                 //1
            System.out.println(h.getAddress());         //2
      }
}
```

Select 1 option:

A. Code for class HomeOffice will cause compilation to fail.

B. Line at //1 will cause compilation to fail.

C. Line at //2 will cause compilation to fail.

D. The code will compile successfully if the getAddress method is removed from class HomeOffice.

E. It will compile fine and print R No 1, Home when run.

Question 33.

Consider these two interfaces:

```
interface I1
{
      void m1() throws java.io.IOException;
}
interface I2
{
      void m1() throws java.io.FileNotFoundException;
}
```

Which of the following are valid method declarations for a class that says it implements I1 and I2 ?

Select 1 option:

A. Both, public void m1() throws FileNotFoundException; and public void m1() throws IOException;

B. public void m1() throws FileNotFoundException

C. The class cannot implement both the interfaces as they have conflicting methods.

D. public void m1() throws Exception;

E. None of the above.

Question 34.

What will happen when the class WaitTest is run using the command line:
java WaitTest a c b

```
import java.util.*;

public class WaitTest {
      static boolean sorted = false;
      public static void main(String[] args) {
            new AnotherThread(args).start();          //1
            synchronized (args) {
                  try {
                        while(!sorted){
                              args.wait();
                        }
                  } catch (InterruptedException e) { }
                  List<String> m = Arrays.asList(args);
                  System.out.println(m);
            }
      }
}

class AnotherThread extends Thread {

      String[] args;

      AnotherThread(String[] sa) {
            args = sa;
      }

      public void run() {
            synchronized (args) {
                  List<String> m = Arrays.asList(args);
                  Collections.sort(m);
                  WaitTest.sorted = true;
                  args.notifyAll();
            }
      }
}
```

Select 2 options:

A. It WILL print [a, b, c].

B. It MAY print [a, c, b].

C. It MAY print nothing.

D. It will NEVER print [a, c, b].

E. It will not compile.

F. It may print the elements of the args array in any order.

G. It will throw an exception at runtime.

Question 35.

Which of the following calls made by the current thread will NOT stop the current thread from executing ?
(Assume that the calls are made from a subclass of Thread class.)

Select 3 options:

A. `start();`

B. `notify();`

C. `wait();`

D. `interrupt();`

E. `t.join();` //here t is a reference to some other thread.

Question 36.

Given the following class, what should be inserted at **//line 1** to make sure that it prints **h.i = 20**?

```
class Hello implements Runnable{
      int i;
      public void run(){
            try {
                  Thread.sleep(3000);
            } catch (InterruptedException e){}
            i = 20;
      }
}

public class Test {
      static public void main(String[] args) throws Exception{
            Hello h = new Hello();
            Thread t = new Thread(h);
            t.start();
            //line 1
            System.out.println("h.i = " + h.i);
      }
}
```

a. h.wait();
b. t.wait();
c. t.yield();
d. t.join();
e. h.notify();
f. t.notify();
g. t.interrupt();

Select 1 option:

A. any one of a, b, g

B. any one of c, e, f

C. any one of c, d, f

D. Only d

E. Only b

Question 37.

What will be the output when the following program is compiled and run?

```
public class TestClass extends Thread {
      String name = "";
      public TestClass(String str) {
            name = str;
      }
      public void run() {
            try {
                  Thread.sleep( (int) (Math.random()*1000) );
                  System.out.println(name);
            } catch(Exception e) { }
      }
      public static void main(String[] str) throws Exception {
            Thread t1 = new TestClass("tom");
            Thread t2 = new TestClass("dick");
            t1.start();t2.start();
            t1.join(); t2.join();
            System.out.println("harry");
      }
}
```

Select 1 option:

A. It will always print tom, dick, and harry, in that order.

B. It will always print harry in the end.

C. It may print tom, dick, and harry, in any order.

D. tom will always be printed before dick.

E. tom will always be printed first.

Question 38.

Given the following class definitions, the expression

```
(obj instanceof A) && ! (obj instanceof C) && ! (obj instanceof D)
```

correctly identifies whether the object referred to by obj was created by instantiating class B rather than classes A, C and D?

```
      class A {}
      class B extends A {}
      class C extends B {}
      class D extends C {}
```

Select 1 option:

A. True

B. False

Question 39.

Given :

```
interface Process{
      public void process(int a, int b);
}

public class Data{
      int value;
      Data(int value){
            this.value = value;
      }
}
```

and the following code fragments:

```
public void processList(ArrayList<Data> dataList, Process p){
      for(Data d: dataList){
            p.process(d.value, d.value);
      }
}

....

      ArrayList<Data> al = new ArrayList<Data>();
      al.add(new Data(1));al.add(new Data(2));al.add(new Data(3));

      //INSERT METHOD CALL HERE
```

Which of the following options can be inserted above so that it will print **1 4 9**?

Select 3 options:

A. processList(al, a, b->System.out.println(a*b));

B. processList(al, (int a, int b)->System.out.println(a*b));

C. processList(al, (int a, int b)->System.out.println(a*b););

D. processList(al, (a, b)->System.out.println(a*b));

E. processList(al, (a, b) ->{ System.out.println(a*b); });

Question 40.

Identify the valid for loop constructs assuming the following declarations:

```
Object o = null;
Collection c = //valid collection object.
int[][] ia = //valid array
```

Select 2 options:

A. `for(o : c){ }`

B. `for(final Object o2 :c){ }`

C. `for(int i : ia) { }`

D. `for(Iterator it : c.iterator()){ }`

E. `for(int i : ia[0]){ }`