1 Suppose we have the following classes in our program. (1)

```
public abstract class Toy { ... }
public class Doll extends Toy { ... }
public class Barbie extends Doll { ... }
```

Assume that `Toy`, `Doll`, and `Barbie` each have constructors that accept no parameters.

If an object, `child`, has a method `play(Doll doll)`, which of the following statements will compile with no errors?

A. `child.play(new Object());`

B. `child.play(new Toy());`

C. `child.play(new Doll());`

D. `child.play(new Barbie());`

2 Indicate whether each of the following statements is `True` or `False.` (1)

A. You can create interfaces that are "subinterfaces" of other interfaces.

B. A class can implement only one interface at a time.

C. An interface can be used as the data type for parameters in methods and while declaring variables.

3 Indicate whether each of the following statements is `True` or `False`. (2)

A. You can't create objects of an abstract class.

B. You can't create non-abstract subclasses of an abstract class.

C. You can't create abstract subclasses of an abstract class.

D. `Object` is an abstract class.

E. An abstract class can be used as the data type for parameters in methods and while declaring variables.

4 Complete the `compareTo()` method below which will determine whether or not an `Atom` object appears before or after another `Atom` in the periodic table. (2)

```
class Atom implements Comparable<Atom> {
  private int atomicNumber;

  public int compareTo(Atom a) {
    //  To be implemented.
  }

  // There may be instance variables, constructors, and other methods not shown.
}
```

5 Answer each of the following questions. (4)

(a) Describe a `BankTransaction` class which would handle the various different types of transactions between accounts in a bank. In particular, it should be able to describe withdrawals, deposits, and tranfers of funds between accounts. How would you design such a class? In particular, describe the instance variables and methods which might exist for this class.

(b) Would it be appropriate to represent `BankTransaction` as a `class`, `abstract class`, or an `interface`? Explain your choice.

6 Complete each of the following exercises and questions. (6)

(a) Implement a `BankAccount` abstract class with:

- an appropriate instance variable to hold the current balance in the account,
- abstract method `processTransaction()` which will take a `BankTransaction` parameter,
- helper methods `depositAmount()` and `withdrawAmount()` which should update the balance in the account appropriately, and
- an accessor method to the balance in the account.

(b) Complete the `makePayment()` method below.
Note: This method would likely be part of a larger system and should *not* be included in the `BankAccount` class.

```java
/**
 * Generates appropriate BankTransaction objects to withdraw money from one
 * account and deposit it into another.
 *
 * @param from  The account to transfer an amount from.
 * @param to    The account to transfer an amount to.
 * @param amt   The amount to transfer.
 * @return  Returns true if the transaction was successful and
 *          false otherwise.
 */
public boolean makePayment(BankAccount from, BankAccount to, double amt) {
  //  To be implemented
}
```

(c) Suppose `CheckingAccount`, `SavingsAccount` and `LoanAccount` were each subclasses of `BankAccount`. Briefly describe how their `processTransaction()` implementations might differ.

(d) Why is it appropriate to implement `BankAccount` as an abstract class rather than an interface?