Questions 4–5 refer to the `Card` and `Deck` classes shown below.

```
public class Card
{
    private String suit;
    private int value;      //0 to 12

    public Card(String cardSuit, int cardValue)
    { /* implementation */ }

    public String getSuit()
    { return suit; }

    public int getValue()
    { return value; }

    public String toString()
    {
        String faceValue = "";
        if (value == 11)
            faceValue = "J";
        else if (value == 12)
            faceValue = "Q";
        else if (value == 0)
            faceValue = "K";
        else if (value == 1)
            faceValue = "A";
        if (value >= 2 && value <= 10)
            return value + " of " + suit;
        else
            return faceValue + " of " + suit;
    }
}

public class Deck
{
    private Card[] deck;
    public final static int NUMCARDS = 52;

    public Deck()
    { ...

    /** Simulate shuffling the deck. */
    public void shuffle()
    { ...

    //Other methods are not shown.
}
```

4. Which of the following represents correct /* *implementation* */ code for the constructor in the `Card` class?

(A)
```
suit = cardSuit;
value = cardValue;
```

(B)
```
cardSuit = suit;
cardValue = value;
```

(C)
```
Card = new Card(suit, value);
```

(D)
```
Card = new Card(cardSuit, cardValue);
```

(E)
```
suit = getSuit();
value = getValue();
```

5. Consider the implementation of a `writeDeck` method that is added to the `Deck` class.

```
/** Write the cards in deck, one per line. */
public void writeDeck()
{
    /* implementation code */
}
```

Which of the following is correct /* ***implementation code*** */?

I
```
System.out.println(deck);
```

II
```
for (Card card : deck)
    System.out.println(card);
```

III
```
for (Card card : deck)
    System.out.println((String) card);
```

(A) I only
(B) II only
(C) III only
(D) I and III only
(E) II and III only

m

6. Refer to the following method that finds the smallest value in an array.

```
/** Precondition: arr is initialized with int values.
 *  @param arr the array to be processed
 *  @return the smallest value in arr
 */
public static int findMin(int[] arr)
{
    int min = /* some value */;
    int index = 0;
    while (index < arr.length)
    {
        if (arr[index] < min)
            min = arr[index];
        index++;
    }
    return min;
}
```

Which replacement(s) for `/*` *some value* `*/` will always result in correct execution of the `findMin` method?

I `Integer.MIN_VALUE`

II `Integer.MAX_VALUE`

III `arr[0]`

(A) I only
(B) II only
(C) III only
(D) I and III only
(E) II and III only

7. Consider the following loop, where `n` is some positive integer.

```
for (int i = 0; i < n; i += 2)
{
      if (/* test */)
            /* perform some action */
}
```

In terms of `n`, which Java expression represents the maximum number of times that `/*` *perform some action* `*/` could be executed?

(A) `n / 2`

(B) `(n + 1) / 2`

(C) `n`

(D) `n - 1`

(E) `(n - 1) / 2`

8. A method is to be written to search an array for a value that is larger than a given item and return its index. The problem specification does not indicate what should be returned if there are several such values in the array. Which of the following actions would be best?

(A) The method should be written on the assumption that there is only one value in the array that is larger than the given item.

(B) The method should be written so as to return the index of every occurrence of a larger value.

(C) The specification should be modified to indicate what should be done if there is more than one index of larger values.

(D) The method should be written to output a message if more than one larger value is found.

(E) The method should be written to delete all subsequent larger items after a suitable index is returned.